NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

DEVELOPMENT AND VALIDATION OF A CONTROLLED VIRTUAL ENVIRONMENT FOR GUIDANCE, NAVIGATION AND CONTROL OF QUADROTOR UAV

by

Junwei Choon

September 2013

Thesis Advisor: Oleg Yakimenko
Thesis Co-Advisor: Isaac Kaminer

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) WashingtonDC20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2013	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE DEVELOPMENT AND VALIDATION OF A CONTROLLED VIRTUAL ENVIRONMENT FOR GUIDANCE, NAVIGATION AND CONTROL OF QUADROTOR UAV		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Junwei, Choon			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.IRB Protocol number: N/A.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200words)

This thesis is focused on the development of a six degrees of freedom (6DOF) simulation model of a commercial-off-the-shelf quadrotor. The dynamics of the quadrotor and its control strategy are described. The Geometric Dilution of Precision (GDOP) of the Autonomous Systems Engineering and Integration Laboratory (ASEIL) laboratory used in conducting the experiments is also analyzed. Simulation results are then verified with actual flight data.

A direct method of calculus of variations is employed in the development of an algorithm for optimal trajectory generation and collision-free flight. Using the differential-flatness characteristics of the system, the trajectory optimization is posed as a nonlinear constrained optimization problem in virtual domain, not explicitly related to the time domain. Appropriate parameterized functions employing an abstract argument, known as the virtual arc, are used to ensure initial and terminal constraints satisfaction. A speed factor maps the virtual to the time domain and controls the speed profile along any predetermined trajectory. An inner loop attitude controller was used to achieve almost global asymptotic attitude tracking for trajectory following. The trajectory generation and following algorithms were verified using the 6DOF simulation model through a simulated collision avoidance scenario.

14. SUBJECT TERMS Quadrotor, Qball, 6DOF Model, PID, Direct Methods, Trajectory, Optimization, Inverse Dynamics, IDVD, Collision Avoidance			**15. NUMBER OF PAGES** 179
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

DEVELOPMENT AND VALIDATION OF A CONTROLLED VIRTUAL ENVIRONMENT FOR GUIDANCE, NAVIGATION AND CONTROL OF QUADROTOR UAV

Junwei Choon
Civilian Engineer, Singapore Technologies Aerospace
B.Eng (Mechanical), Nanyang Technological University, Singapore, 2008

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATESCHOOL
September 2013

Author: Junwei Choon

Approved by: Oleg Yakimenko
Thesis Advisor

Isaac Kaminer
Thesis Co-Advisor

Knox Millsaps
Chair, Department of Mechanical and Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis is focused on the development of a six degrees of freedom (6DOF) simulation model of a commercial-off-the-shelf quadrotor. The dynamics of the quadrotor and its control strategy are described. The Geometric Dilution of Precision (GDOP) of the Autonomous Systems Engineering and Integration Laboratory (ASEIL) laboratory used in conducting the experiments is also analyzed. Simulation results are then verified with actual flight data.

A direct method of calculus of variations is employed in the development of an algorithm for optimal trajectory generation and collision-free flight. Using the differential-flatness characteristics of the system, the trajectory optimization is posed as a nonlinear constrained optimization problem in virtual domain, not explicitly related to the time domain. Appropriate parameterized functions employing an abstract argument, known as the virtual arc, are used to ensure initial and terminal constraints satisfaction. A speed factor maps the virtual to the time domain and controls the speed profile along any predetermined trajectory. An inner loop attitude controller was used to achieve almost global asymptotic attitude tracking for trajectory following. The trajectory generation and following algorithms were verified using the 6DOF simulation model through a simulated collision avoidance scenario.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ABC	Aircraft Body Coordinates
ASEIL	Autonomous Systems Engineering and Integration Laboratory
COTS	Commercial-Off-the-Shelf
DAC	Data Acquisition Card
DCM	Directional Cosine Matrix
EOD	Explosive Ordnance Disposal
ESC	Electronic Speed Controller
ENU	East-North-Up
GPS	Global Positioning System
IMU	Inertial Measurement Unit
I/O	Input/Output
IED	Improvised Explosive Device
IDVD	Inverse Dynamics In Virtual Domain
LTP	Local Tangent Plane
MAV	Micro Air Vehicles
MOI	Moments Of Inertia
NLP	Nonlinear Programming
ODE	Ordinary Differential Equation
PID	Proportional Integral Derivative
PWM	Pulse Width Modulation
QuarC	Quanser Real-time Control
RC	Remote Controlled
RPV	Remotely Piloted Vehicles
SLAM	Simultaneous Localization and Mapping
UAV	Unmanned Air Vehicle
USB	Universal Serial Bus
VTOL	Vertical Takeoff and Landing
6DOF	Six Degrees-0f-Freedom

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

To find what you seek in the road of life, the best proverb of all is that which says:

"Leave no stone unturned.”

- Henry David Thoreau

Looking back on my experience as a master’s candidate, only now do I realize that it is more like a journey than the end result. In that respect, I feel an even greater sense of achievement and growth than I do for the results in this thesis. I owe this accomplishment to the fact that I had an extraordinary thesis advisor, Prof Oleg Yakimenko and continuous strong support from Prof Issac Kaminer and Prof Vladimir Dobrokhodov. Prof Oleg Yakimenko, your tireless effort and attention to details inspired me and gave me direction.

I would like to thank Prof Isaac Kaminer for his patience and guidance. Special thanks to Prof Vladimir Dobrokhodov for putting his best effort in teaching me in the area of controls engineering.

Appreciation is also extended to lab technicians Juan Gonzales and Tommy Pierce for their technical support during the course of my studies and to Cameron Fulford for providing valuable information on the Quanser Qball-X4.

Last but not least, I want to thank my parents for their love and support. Without them, I could never have accomplished so much.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

Recent advances in miniature technology have brought a global spotlight on the development of Unmanned Aerial Vehicles (UAVs), also commonly known as drones or remotely piloted vehicles (RPVs). A UAV is defined as being capable of controlled, sustained level flight and powered by a jet or reciprocating engine driving a propeller. Electric, battery or fuel-cell powered motors are becoming usual on micro- and mini-UAVs (Kansas State University 2012).

UAVs are mostly assigned to dull, dirty and dangerous missions, therefore preventing the exposure of humans to uncertain or hostile environments that can potentially pose a danger to the lives of operators. For instance, Honeywell's T-Hawk RQ-16, an autonomous micro air vehicle (MAV) with hover and stare capabilities had been deployed in Afghanistan to assist ground troops in counter improvised explosive devices (IEDs) activities, as well as in combat deployment in Iraq as Explosive Ordnance Disposal (EOD) UAVs (Croft 2010). The MAV also demonstrated its role in civilian applications when it was used to conduct surveillance of the damaged Fukushima Dai-Ichi nuclear power station in 2011 (Net Resources International 2012). These are just a few of the many examples of what UAVs are capable of doing.

Among UAVs, those with rotary-wing configurations have advantages over small fixed-wing UAVs since they can hover in place and are more maneuverable, and they can be launched by an operator staying behind cover while small

fixed-wing drones have to be hand-launched with the operator either standing or running. Thus, rotorcrafts can be deployed in a much wider range of scenarios.

Among the different types of rotorcrafts, quadrotors have been particularly popular in the research field since early 2000s. There have been many publications [4-9] in recent years describing the dynamics and controls of quadrotors. Quadrotors are mechanically simple and can be controlled only by changing the rotational speed of the four rotors. They are highly agile, but the dynamics of the quadrotors can make them difficult to control; thus electronic stability augmentation is usually required for stable flight (Hanford, Long and Horn 2005).

Direct methods for real-time trajectory generation and trajectory following for UAVs are crucial for time-critical, collision-free flight during a mission with a single UAV or cooperative control of multiple UAVs. Real-time trajectory generation and following are motivated by applications of precision control. While computing optimal trajectories can be a complicated matter, there are many situations for which nothing less will solve the problem. This is evident during obstacle avoidance in close proximity and during formation flight by a swarm of UAVs.

Motivated by these challenges, the objective of this thesis is comprised of two parts. First, the development of a six degrees of freedom (6DOF) model with animation for visualization of a commercial-off-the-shelf (COTS) quadrotor in the Simulink environment is explored. This allows testing of algorithms on the simulation model prior to implementation, testing and verification on the actual

platform itself. Second, this thesis examines the implementation of the direct method of calculus of variations exploiting the inverse dynamics of the quadrotor in virtual domain (IDVD) for optimal trajectory generation and employing a nonlinear attitude tracking controller for trajectory following. These algorithms are verified through a collision avoidance mission scenario.

B. WHY USE QUADROTORS?

1. ADVANTAGES

There are several advantages associated with the use of quadrotors compared to small fixed-wing UAVs. A few of the advantages are listed here.

Hover-Capable. Unlike conventional fixed-wing UAVs, quadrotors have the ability to hover in place over an extended duration. This gives quadrotors an advantage over fixed-wing UAVs when a mission requires persistent monitoring of a target. The ability to hover allows the vehicle to vertically takeoff and land (VTOL), thus minimizing the footprint needed to launch and land the vehicle and preventing exposure of human operators to possible dangers. Use of VTOL-capable UAVs also eliminates the need for any launch and recovery equipment, thus simplifying the logistics required for operating such systems.

Highly Maneuverable. Quadrotors are highly agile. They can execute sharp turns almost instantaneously compared to fixed-wing UAVs, which have a much larger turning radius. Quadrotors are also better suited to operating in indoor

environments where room for maneuvering can be very restrictive.

Small Size. Current miniature technology facilitates the construction of micro-size quadrotors. This allows them to be easily transported and deployed. Being small in size also means that they can easily fit through tight windows and doorways. The survivability of such vehicles is also increased since they are less likely to be detected by enemy forces when deployed in tactical missions.

Mechanically Simple. Unlike conventional helicopters, quadrotors do not require mechanically complicated variable-pitch mechanisms for their rotors. Instead, they generally employ symmetrically pitched blades. Therefore, they only have a minimum number of moving parts. Maneuvering of the vehicle is accomplished merely by changing the rotational speed of the rotors.

2. DISADVANTAGES

Like all systems, quadrotors also have their disadvantages. Some of the disadvantages are listed here.

Low Endurance. Fixed-wing UAVs leverage the air flowing across the wings to generate aerodynamic lift, while quadrotors have to carry their own weight via the thrust generated by the rotors. Thus, quadrotors consume more power to stay aloft than their fixed-wing UAV counterparts. With the current level in battery technology, the maximum endurance of quadrotors is typically less than an hour.

Limited Payload. Quadrotors generally have payload restrictions which limit the size and the number of sensors they can carry onboard.

C. THESIS OUTLINE

This section presents the outline for this thesis.

Chapter I includes the background information and motivation for this thesis.

Chapter II provides the literature review, and describes several direct and indirect methods in optimal control theory. Several related projects by universities are also described in this chapter.

Chapter III describes the modeling of the quadrotor's dynamics. This involves deriving the equations of motion for the development of a complete 6DOF simulation model.

Chapter IV describes the implementation of the 6DOF simulation model in Simulink and its interface with the controllers used for controlling the quadrotor (Qball-X4) used in the work of this thesis.

Chapter V provides an analysis of the dilution of precision at different locations in the ASEIL laboratory used for conducting the experiments.

Chapter VI compares the results from the simulation model to the actual flight data.

Chapter 0 introduces the direct method of calculus of variations exploiting IDVD for optimal trajectory generation and trajectory following.

Chapter VIII demonstrates the application of the IDVD method in the 6DOF simulation model through a collision avoidance scenario and presents the results.

Chapter IX highlights the conclusions drawn from the research and recommendations for future work.

II. LITERATURE REVIEW AND RECENT WORKS

A. LITERATURE REVIEW

In recent years, many researchers have addressed the control problem associated with quadrotors [4-19]. Typical control of quadrotors includes attitude stabilization and movement from one pose (position and attitude) to another.

Dynamic modeling of quadrotors has been performed by many researchers [20-23]. Dynamic modeling of Draganflyer XP, a commercial quadrotor, was proposed by Bradford et al. (Bradford, Nelson and Palm 2010). Bristeau et al. published a paper describing the navigation and control technology inside the AR.Drone quadrotor (Bristeau, et al. 2011). Several researchers have also used the Qball-X4 quadrotor as the testbed for their algorithms [24,25].

Proportional, Integral and Derivative (PID) control, a technique developed in the 1890s (Bennett 1993), for controlling a quadrotor were studied by Szafranski and Czyba (Szafranski and Czyba 2011), Bouabdallah et al. (Bouabdallah, Noth and Siegwart 2004) and Salih et al. (Salih, et al. 2010). Nonlinear control problems for hovering quadrotors such as feedback linearization control and backstepping control laws were investigated by Altug et al. (Altug, Ostrowski and Mahony 2002) and Madani and Benallegue (Madani and Benallegue 2007). Recently in 2012, Serirojanakul (Serirojanakul 2012) suggested using state feedback linear parameter-varying (LPV) method for optimal control of a quadrotor. The nonlinear model of the quadrotor is first transformed into a linear model subjected to time-varying parameters; then the composite

quadratic Lyapunov function and quadratic cost functions are used to find the optimal state feedback gain.

Stability of an inherently instable quadrotor is always a concern to researchers. A hybrid backstepping control and the Frenet-Serret theory used for attitude stabilization was proposed by Colorado and Barrientos (Colorado and Barrientos 2010) while Coza (Coza 2006) implemented a robust adaptive-fuzzy control method to stabilize a quadrotor. Shepherd and Tumer (Shepherd and Tumer 2010)used a hierarchical neuro-controller to stabilize flight of a micro quadrotor in the presence of five times more sensor noise and eight times more actuator noise compared to the PID controller.

There is a growing interest in UAVs acquiring an increased level of autonomy as more complex mission scenarios are envisioned (Office of the Secretary of Defense 2005). This interest has inspired many researchers to develop algorithms for the optimal control of quadrotors in a nondeterministic environment. The Linear Quadratic Regulator (LQR) is one type of optimal control technique that constructs a control law in order to minimize a cost function in which the required state feedback matrix must be known. LQR is applied to the quadrotor by casting the differential equations describing the model into state-space form, transforming all the differential equations into a first order system (Nuchkrua and Parnichkun 2012). The nonlinear matrix algebraic Riccati equation is solved for obtaining optimal feedback gain matrices. The disadvantage of those methods is the complexity in computing the matrix algebra in a digital computer

processor for real-time applications. As such, researchers have developed techniques to overcome this difficulty. Yakimenko developed algorithms using the direct method of calculus of variations exploiting the inverse dynamics in virtual domain that are capable of generating near optimal trajectories in real-time [11,27-28]. Cowling and Yakimenko (Cowling, Yakimenko, et al. 2010) tested this method on an autonomous quadrotor. Hehn and D'Andrea (Hehn and D'Andrea 2012) also developed algorithms for real-time trajectory generation for interception maneuvers with quadrotors. Mellinger and Kumar (Mellinger and Kumar 2011) developed an algorithm that enables the real-time generation of optimal trajectories through a sequence of 3D positions and yaw angles for an aggressive maneuvering quadrotor. Their optimization approach minimizes the cost functional derived from the square of the norm of the snap (fourth derivative of position).

At this point, it is convenient to review some of the optimal control methods, generally classified as either a direct or indirect method. Researchers who focus on indirect methods are largely interested in differential equation theory while researchers who focus on direct methods are more interested in optimization techniques (Rao 2009). This difference in methods will be discussed in the next section.

B. BACKGROUND IN OPTIMAL CONTROL

Finding the best way for a quadrotor to get from one place to another can be described as an optimal control problem, while optimal control problems are generally nonlinear and, therefore, do not have analytic solutions.

It is necessary to employ numerical methods to solve for optimal control problems. When describing methods for solving optimal control problems, a technique is often classified as either a direct or an indirect method. In the early years of optimal control (circa 1950s to 1980s) the favored approach for solving optimal control problems was that of indirect methods. However, the disadvantage associated with indirect methods is that the boundary-value problem is often extremely difficult to solve. In recent decades, direct methods are becoming more popular. The nonlinear programming (NLP) problems arising from direct methods are usually easier to solve compared to boundary-value problems. The approach used by direct and indirect methods is described in Figure 1.

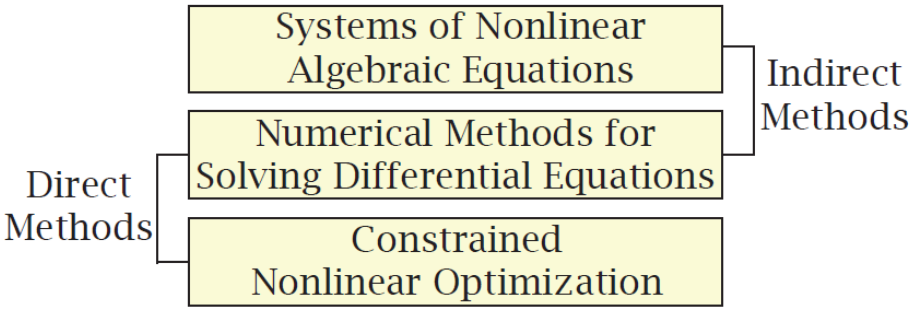


Figure 1: Approach in Direct and Indirect Methods (From Rao 2012).

It should be noted that direct methods are the more practical ones for real-time applications since indirect methods generally take impractically long to find a valid solution if that is even possible.

C. INDIRECT METHODS

Indirect methods seek a solution to the (closed system of) necessary conditions of optimality. Discretization of

the control profile is not needed, but it requires a guess on the optimal solution structure which is often non-intuitive. It is necessary to derive the adjoint equations, control equations and all the transversality conditions explicitly. Thus, solving optimal control problems using indirect methods becomes a root-finding problem. The optimality conditions are also often not trivial to formulate. Examples of indirect methods include the gradient method and the multiple shooting method, which are discussed in the following sections.

1. Gradient Method

The gradient method to solve for optimal control problems was first introduced by Lasdon et al. (Lasdon, Mitter and Waren 1967). The search directions used in this method are generated from the past and present values of the objective and its gradient. Iterations using linear minimization are always in the direction of descent; thus this method tends to converge even from poor approximations to the minimum. The advantage of the gradient method is that each iteration is inexpensive and does not require second derivatives; however, this method is often slow in attaining convergence and is scaling dependent, such that the number of iterations largely depends on the scale of the problem. The gradient method also cannot solve for non-differentiable problems; however, there are various enhancements to the gradient method to address these drawbacks. To improve convergence, techniques such as variable metric methods (Turner and Huntley 1980), conjugate gradient methods (Lasdon, Mitter and Waren 1967) and accelerated gradient methods (Cotter, et al. 2011) were

used. To overcome non-differentiable or constrained problems, sub-gradient methods (Shor 1985), proximal gradient methods (Chen and Ozdaglar 2012), smoothing methods (Chen, Nashed and Qi 2000) and cutting-plane methods (Elhedhli, Goffin and Vial 2009) were also used.

2. Multiple Shooting Method

The multiple shooting method has proved to be an effective tool in solving highly nonlinear multi-point boundary value problems. This method is described by, for example, Stoer and Bulirsch (Stoer and Bulirsch 1980). Shooting refers to a strategy for finding unknown parameters, primarily the initial values of variables. A trial shot is made at solving the necessary conditions, primarily the multi-point boundary conditions, by integrating the equations with guessed parameters. Then, the shot is adjusted iteratively by varying the parameters, until the adjusted shot satisfies the necessary conditions. The indirect shooting method is depicted using an analogous illustration of a cannon shooting at a target in Figure 2.

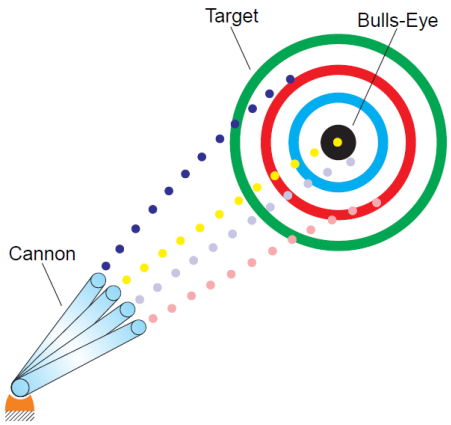


Figure 2: Schematic of Indirect Shooting Method Using the Analogy of a Cannon Firing a Cannonball to Strike a Target (From Rao 2009).

The major advantage of the multiple shooting method is its potential to obtain a highly accurate solution through the verification of the optimality conditions. The main drawbacks include the necessity to derive the necessary conditions (e.g., the adjoint differential equations), guess the optimal switching structure and make an appropriate initial estimate of the unknown state and adjoint variables in order to start the iteration process (Stryk 1996).

D. DIRECT METHODS

In direct methods, the optimal control problem is first discretized. Then NLP techniques are applied to the resulting finite-dimensional optimization problem. The state and control can be approximated using suitable function approximations, such as a polynomial approximation or piecewise constant parameterization. This leads to a finite number of unknown coefficients that are defined by the variation principles, boundary-value conditions and collocation requirements, which need to be determined.

The advantage of direct methods is that a priori knowledge of the solution structure is not required; however, they offer only an approximate solution due to control parameterization. Direct methods can generally be classified according to Figure 3.

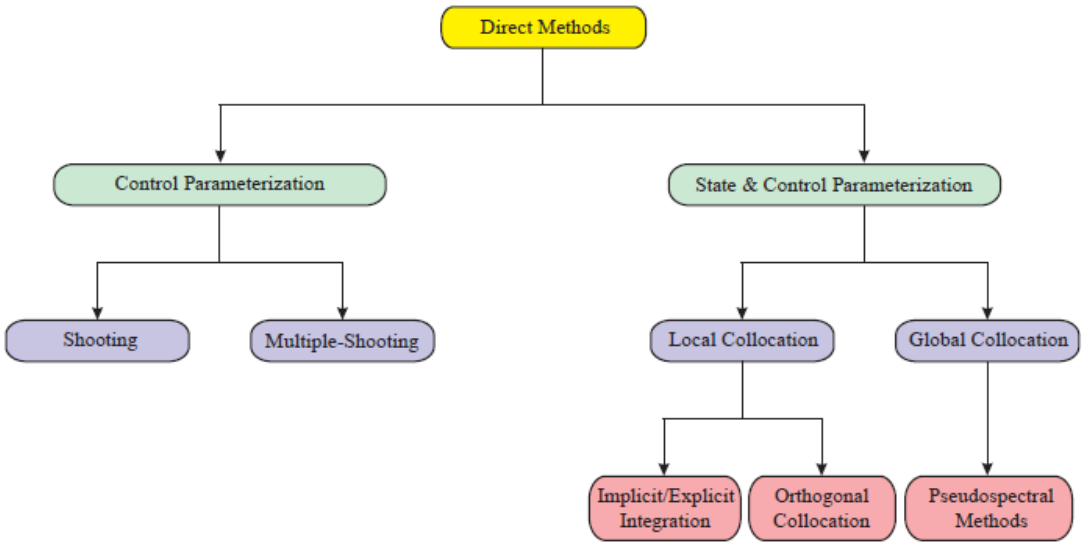


Figure 3: Different Types of Direct Methods (From Rao 2009).

1. Direct Transcription Method

In a direct transcription method, the dynamic system is transcribed into a problem with a finite set of variables. The finite dimensional problem is then solved using a parameter optimization method (i.e., the NLP sub-problem). The accuracy of the finite dimensional problem is then assessed, and transcription and optimization steps are repeated, if necessary. This method is described by Engelsone (Engelsone 2006) and Betts (J. T. Betts 2001).

2. Inverse Dynamics Approach

Inverse dynamics is a design technique whereby the set of existing or undesirable dynamics of a system are eliminated and replaced by a designer selected set of desired dynamics. An illustration of this concept is given in Figure 4.

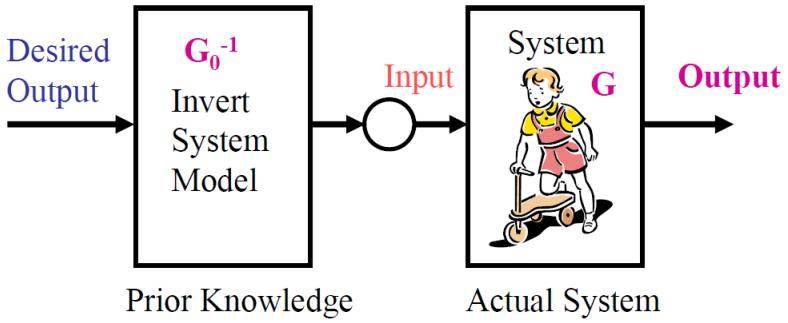


Figure 4: Illustration of the Inverse Dynamics Approach (From Devasia n.d.).

An advantage of using the inverse dynamics-based dynamic programming method over conventional dynamic programming methods is the elimination of the interpolation requirement for systems. This requirement can degenerate the accuracy due to errors associated to the interpolation process. This method is used by Chen and Tsong (Chen and Tsong 1998) to solve for optimal control problems of linear systems.

3. Pseudospectral Methods

In pseudospectral methods (PM), the continuous functions are approximated at a set of carefully selected quadrature nodes. The quadrature nodes are determined by the corresponding orthogonal polynomial basis used for the

approximation. Theoretically, quadrature nodes are capable of attaining high accuracy with a small number of points.

The main appeal of the PM is its exponential (or spectral) rate of convergence, which is faster than any polynomial rate, and the possibility to achieve good accuracy with relatively coarse grids.

These methods include forms of the collocation at the Legendre-Gauss-Lobatto (LGL) points (Gong, Kang and Ross 2006), collocation at Chebyshev-Gauss-Lobatto (CGL) points (Elnagar and Kazemi 1998), Legendre-Gauss points (LG) (Benson, et al. 2006) and collocation at Legendre-Gauss-Radau points (LGR) (Garg, et al. 2011). Two PM methods using the LG and LGR collocation aim at solving infinite-horizon (i.e., the final time lies in an infinite duration from the actual horizon at $t = +\infty$) optimal control problems were suggested by Garg et al. (Garg, Hager and Rao 2011)

E. RELATED WORK

In recent years, many universities have been using quadrotors as the testbed for their new ideas in a number of fields, including flight control, navigation and real-time systems. The cross-fertilization of ideas and approaches that these projects generate can bring considerable benefits.

1. University of Pennsylvania

Perhaps the most astounding demonstrations of quadrotors come from the General Robotics, Automation, Sensing and Perception (GRASP) at the University of Pennsylvania (Upenn). Videos show quadrotors hovering in mid-air, flying in formation before autonomously performing

complex flying routines like flips, darting through hoops thrown into the air and organizing themselves to fly through windows as a group. A latest video also demonstrated a team of quadrotors playing musical instruments.

An external localization system (VICON) comprised of 20 infrared sensing cameras and onboard inertia measurement unit was used to facilitate these high precision maneuvers.

Figure 5: Composite Image of a Single Quadrotor Flying through a Thrown Circular Hoop (From Mellinger and Kumar 2011).

Figure 6: Composite Image of a Single Quadrotor Quickly Flying through Three Static Circular Hoops (From Mellinger and Kumar 2011).

2. Stanford University

Stanford University developed its own Testbed of Autonomous Rotorcraft for Multi-Agent Control, known as STARMAC. The STARMAC quadrotor is shown performing an autonomous hover at a waypoint in Figure 7.

Figure 7: STARMAC Quadrotor Developed by Stanford University (From Hoffmann and Waslander 2008).

The STARMAC was also used as the testbed for an autonomous trajectory tracking algorithm through cluttered environments. The tracking controller decouples the path planning from the update rate of the control input. By accepting as inputs a path of waypoints and desired velocities, the control input can be updated frequently to accurately track the desired path, while the path planning occurs as a separate process on a slower timescale.

The trajectory tracking algorithms are space-indexed rather than time-indexed, enforcing the requirement that the predetermined obstacle-free path be tracked without

deviation. The STARMAC platform is capable of path tracking with an indoor accuracy of 10 cm and an outdoor accuracy of 50 cm (Hoffmann and Waslander 2008).

3. Massachusetts Institute of Technology

A variable-pitch quadrotor capable of aggressive aerobatic maneuvers (Figures 8 and 9) was developed by Massachusetts Institute of Technology (MIT). In comparison to typical fixed-pitch quadrotors, their variable-pitch quadrotor has a higher control bandwidth. An optimal algorithm based on Rapidly Expanding Random Trees (RRT*) that offers asymptotical optimality guarantees for trajectories while giving probabilistic completeness was tested on the variable-pitch quadrotor, together with a control law that tracks the reference position trajectories that are smooth through the third derivative (Chaudhari 2011).

Figure 8: Variable-Pitch Quadrotor Developed by MIT (From Cutler 2011).

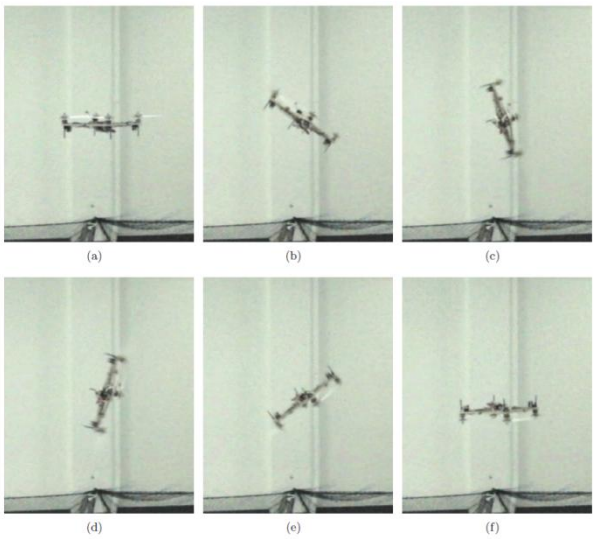


Figure 9: Variable-Pitch Quadrotor Performing 180 degree Flip (From Cutler and How 2012).

4. Naval Postgraduate School

At the Naval Postgraduate School, two AR Drone quadrotors were tasked to follow off-line computed predefined paths, while coordinating their position and attitude according to the scenario requirements. The path tracking controller makes each quadrotor converge and follow its own path independent of the temporal assignments of the scenario (Figure 10). The algorithm relies on the implementation of a virtual vehicle running along the path, synchronizing its position along the path as well as its attitude. Heading can also be controlled independently (Naval Postgraduate School 2013). Localization was also achieved through the external Vicon Motion Tracking system.

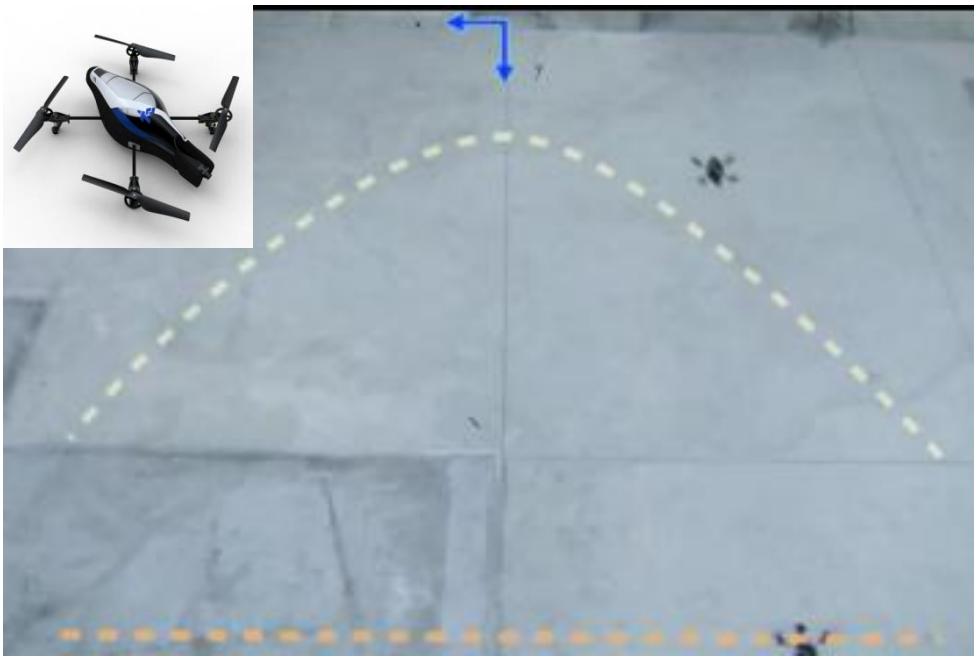


Figure 10: Trajectory Following by Two Parrot AR Drone Quadrotors (After Naval Postgraduate School 2013).

5. Cranfield University

Cranfield University employed the Model-Based Predictive Control (MBPC) technique for combined trajectory planning and following for a quadrotor (Cowling, Whidborne and Cooke 2006). MBPC is a process of repeated optimizations, at every time step, over a fixed finite time horizon to determine the control action, while a control law is determined on-line allowing for improved handling of constraints imposed on the state, inputs and outputs. The real-time trajectory planning allows continual adaptation to a changing environment.

THIS PAGE INTENTIONALLY LEFT BLANK

III. MODELING OF QUADROTOR DYNAMICS AND CONTROL

A. OVERVIEW

The Qball-X4 is a COTS quadrotor helicopter developed by Quanser Consulting, Inc. It is designed mainly for academic research purposes, but it has the potential for commercial applications as well. It is equipped with four standard remotely controlled (RC) motors and electronic speed controllers (ESC), fitted with 10-inch propellers. The quadrotor is enclosed within a protective carbon fiber cage to ensure safe operation to the vehicle and protection to the personnel working with the vehicle in an indoor environment. The Qball-X4 employs fixed-pitched blades. It is equipped with a Quanser Embedded Control Module (QECM), which is comprised of the HiQ data acquisition card (DAQ) and a QuaRC-powered Gumstix embedded computer. A block diagram of the Qball-X4 system is shown in Figure 11.

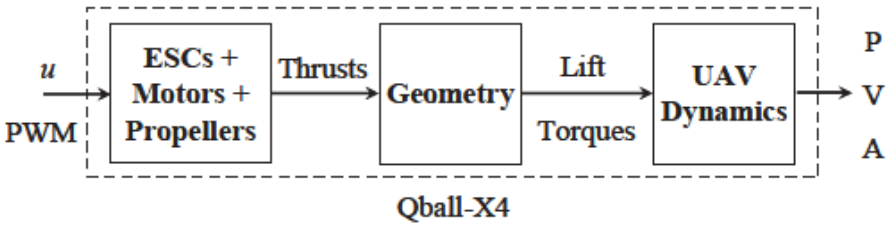


Figure 11: Qball-X4 System Block Diagram (From Zhang and Chamseddine 2012).

This chapter presents the modeling of the dynamics and controller design for the quadrotor. The equations of motion derived are used to construct the 6DOF simulation model, which is the topic of Chapter IV .The mass property

and thrust characteristics of the rotors were obtained through theoretical calculations and experiments. Several simplifying assumptions were also made.

B. DEFINITION OF AXIS SYSTEMS

Two coordinate systems were adopted for the development of the equations of motion and controller design for the quadrotor, namely the Local Tangent Plane (LTP), which in this case is also the Earth Inertial Frame (East-North-Up, ENU frame) and the Aircraft Body Coordinates (ABC) frame. The ENU frame assumes that the Earth is flat, with the x-axis pointing North, the y-axis pointing West and z-axis pointing Up. The flat Earth assumption is valid since the operating workspace is small $(5.5\,m \times 3.5\,m)$ and the duration of the flight is short, i.e., $(< 20\,\text{min})$. The reference origin is taken to be at the center of the workspace.

The Optitrack motion capture system, however, adopted a different coordinate frame as shown in Figure 12, where the x-axis is pointing East, y-axis pointing Up and z-axis pointing South. We will call this frame the Optitrack coordinate frame to avoid confusion. The ENU frame can be readily transformed to the Optitrack coordinate frame using an appropriate transformation matrix which will be given in Section D.

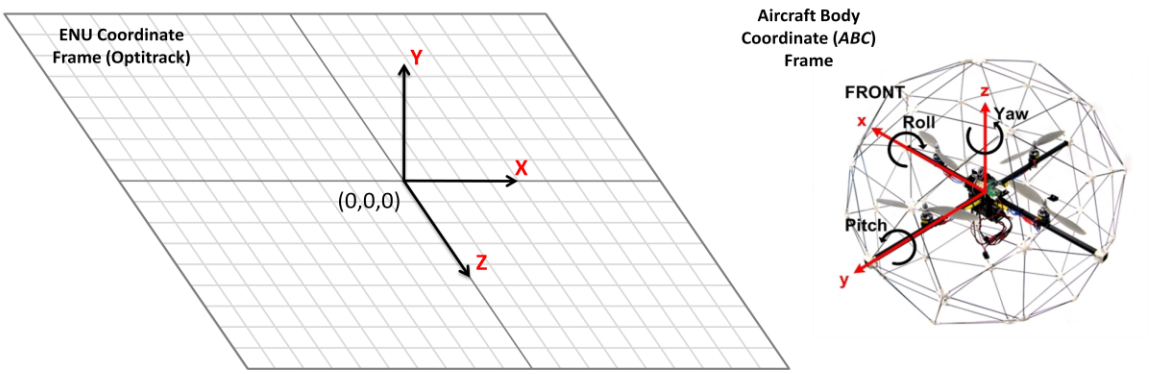


Figure 12: Definition of Axis Systems.

C. ATTITUDE REPRESENTATION

The sequence of rotation conventionally used to describe the instantaneous attitude of the aircraft with respect to the ENU frame is as follows, with the positive Euler angles (Yaw ψ, Pitch θ and Roll ϕ) determined using the Right-Hand Rule:

- Rotate about the body $z-$axis, front beam points left (positive yaw ψ).
- Rotate about the new body $y-$axis, front beam points down (positive pitch θ).
- Rotate about the new body $x-$axis, right beam points down (positive roll ϕ).

It is also noted that from the first rotation step above, the yaw angle is assumed to be the same as the heading angle used for navigation purposes.

D. COORDINATE TRANSFORMATIONS

1. ENU to ABC Transformation

The complete transformation from the ENU frame to the ABC frame is given by the following transformation matrix, commonly referred to as the Directional Cosine Matrix (*DCM*) or the *B*-matrix:

$$B_{ENU2ABC} = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \tag{1}$$

where the elements are given by:

$$\begin{aligned} b_{11} &= \cos\theta\cos\psi \\ b_{12} &= \cos\theta\sin\psi \\ b_{13} &= -\sin\theta \\ b_{21} &= -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi \\ b_{22} &= \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi \\ b_{23} &= \sin\phi\cos\theta \\ b_{31} &= \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ b_{32} &= -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ b_{33} &= \cos\phi\cos\theta \end{aligned}$$

Conversely, the rotational matrix from ABC frame to ENU frame is given by the inverse of the above matrix, such that $B_{ENU2ABC}^{-1}$ or $B_{ABC2ENU}$. Since the coordinate frames are orthogonal, $B_{ENU2ABC}^{-1} \equiv B_{ENU2ABC}^{T}$.

2. ENU to Optitrack Coordinates Transformation

The transformation matrix from ENU frame to the Optitrack coordinates frame is given by:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{Optitrack} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{ENU} \quad (2)$$

E. AIRCRAFT VARIABLES

Table 1 toTable 3 list the aircraft variables used in the equations of motion with the corresponding nomenclature:

Table 1: Angles, Angular Rates and Moments.

Parameter	Nomenclature
Roll Angle in *ENU* Frame	ϕ
Pitch Angle in *ENU* Frame	θ
Yaw Angle in *ENU* Frame	ψ
Angular Rate along Body $x-$axis	p
Angular Rate along Body $y-$axis	q
Angular Rate along Body $z-$axis	r
Rolling Moment	L
Pitching Moment	M
Yawing Moment	N

Table 2: Position Variables.

Parameter	Nomenclature
North Position in *ENU* Frame	X
West Position in *ENU* Frame	Y
Up Position in *ENU* Frame	Z

Table 3: Velocity and Acceleration Variables.

Parameter	Nomenclature
Velocity North in *ENU* Frame	$\dot{X}$
Velocity West in *ENU* Frame	$\dot{Y}$
Velocity Up in *ENU* Frame	$\dot{Z}$
Forward Velocity along Body $x-$axis	u
Lateral Velocity along Body $y-$axis	v
Upward Velocity along Body $z-$axis	w
Acceleration North in *ENU* Frame	a_x
Acceleration West in *ENU* Frame	a_y
Acceleration Up in *ENU* Frame	a_z

F. SIGN CONVENTION FOR PROPELLER ROTATION

The motors and propellers are configured in such a way that the rear and front (1 and 2) motors spin counter-clockwise, and the left and right (3 and 4) spin clockwise as shown in Figure 13. Each motor is located at a distance l

from the center of gravity (CG) of the quadrotor, and τ_i refers to the torque generated by the i^{th} rotor.

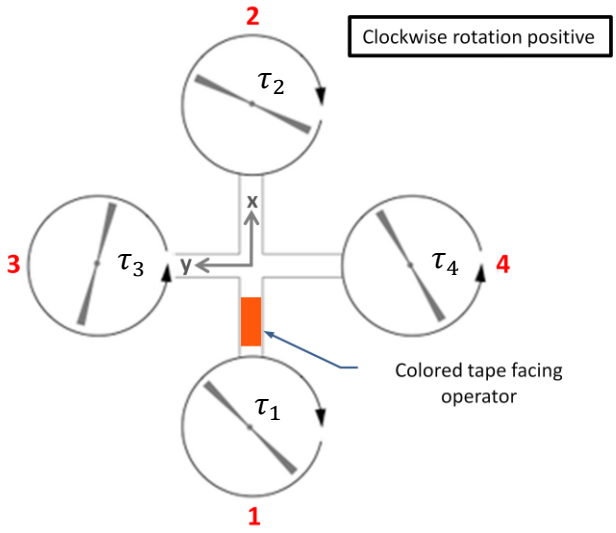


Figure 13: Sign Convention for Rotor Spin Direction.

G. ASSUMPTIONS

Several assumptions concerning the modeling of the quadrotor have been made.

- Flat Earth approximation and non-rotating Earth are assumed. These assumptions are valid since the operating workspace is small and duration of flight is short.
- Gravitational acceleration, $g = 9.81\, ms^{-2}$, is constant and directed along the negative $z-$axis of the ENU frame.
- The quadrotor design is symmetrical about the $xz-$plane and $yz-$plane.
- The quadrotor body and rotor blades are treated as rigid bodies.

- Small angle approximation is used since the quadrotor is maneuvering near to hovering conditions.
- Aerodynamic drag is negligible since the speed is low. The effects of wind, including the ground and wall effects due to the reflected wind from the spinning of the propellers are also neglected.

H. EQUATIONS OF MOTION

This section describes the development of the 6DOF nonlinear aircraft model for the quadrotor. The 6DOF equations of motion are driven by forces and moments from the thrust and torque contribution of the four rotors, acting at the CG of the rigid aerial vehicle. Forces are given the notation F. Rolling, pitching and yawing moments have the notations L, M and N, respectively.

1. Thrust Forces

The thrust forces acting on the CG of the quadrotor are given as:

In ABC frame,

$$F_T^{ABC} = \begin{bmatrix} F_{Tx} \\ F_{Ty} \\ F_{Tz} \end{bmatrix}_{ABC} = \begin{bmatrix} 0 \\ 0 \\ F_{T1} + F_{T2} + F_{T3} + F_{T4} \end{bmatrix} \tag{3}$$

In ENU frame,

$$F_T^{ENU} = \begin{bmatrix} F_{T_x} \\ F_{T_y} \\ F_{T_z} \end{bmatrix}_{ENU} = B_{ABC2ENU} \begin{bmatrix} 0 \\ 0 \\ F_{T1} + F_{T2} + F_{T3} + F_{T4} \end{bmatrix} = B_{ABC2ENU} \begin{bmatrix} 0 \\ 0 \\ F_{total} \end{bmatrix}$$

$$= \begin{bmatrix} F_{total}\sin\phi\sin\psi + F_{total}\cos\phi\sin\theta\cos\psi \\ -F_{total}\sin\phi\cos\psi + F_{total}\cos\phi\sin\theta\sin\psi \\ F_{total}\cos\phi\ cos\theta \end{bmatrix} \quad (4)$$

where F_{Tx}, F_{Ty} and F_{Tz} are the thrust forces acting in the respective coordinate frames. The subscripts ENU and ABC refer to the frame in which the thrust forces are acting. F_{Ti} $(i = 1,2,3,4)$ is the thrust force generated by the i^{th} rotor.

2. Gravity

The forces due to gravity acting in the ENU frame are given as:

$$F_G{}^{ENU} = \begin{bmatrix} F_{Gx} \\ F_{Gy} \\ F_{Gz} \end{bmatrix}_{ENU} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \quad (5)$$

where m is the mass of the quadrotor, and g is the gravitational acceleration.

3. Total Force

The total force acting on the CG of the quadrotor in the ENU frame is given by the sum of the thrust and gravitational forces, while neglecting drag forces.

$$F_{ENU} = F_T{}^{ENU} + F_G{}^{ENU}$$

$$= \begin{bmatrix} F_{total}\sin\phi\sin\psi + F_{total}\cos\phi\sin\theta\cos\psi \\ -F_{total}\sin\phi\cos\psi + F_{total}\cos\phi\sin\theta\sin\psi \\ F_{total}\cos\phi\ cos\theta - mg \end{bmatrix} \quad (6)$$

4. Moments

Roll, pitch and yaw moments are induced by the differential torque generated by the four rotors. To induce a rolling moment, the rotational speeds of rotors 3 and 4 (refer to Figure 13) are varied. To induce a pitching moment, the rotational speeds of rotors 1 and 2 are varied, and finally the rotational speeds of all four rotors contribute to yawing moment.

$$\begin{aligned} &\text{Rolling moment:} && L = \tau_3 - \tau_4 = \left(F_{T3} - F_{T4}\right) l \\ &\text{Pitching moment:} && M = \tau_3 - \tau_4 = \left(F_{T1} - F_{T2}\right) l \\ &\text{Yawing moment:} && N = \left(F_{T3} + F_{T4} - F_{T1} - F_{T2}\right) d \end{aligned} \tag{7}$$

where d is the force-to-moment scaling factor calculated to be 4 Nm, and l is the length of the moment arm measured from the rotor to the quadrotor's CG.

5. Moments of Inertia

The moment of inertia (MOI) about the body axes can be calculated, assuming the mass contributions mainly come from the central airframe body and the four motors of the quadrotor, and that they assume the shape of solid cylinders as shown in Figure 14.

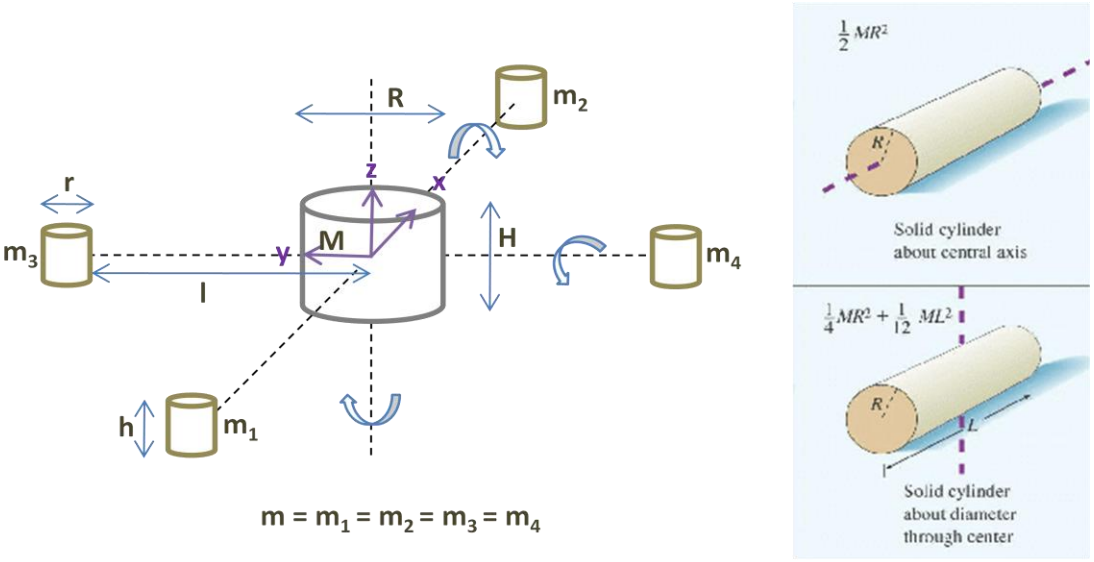


Figure 14: Calculating the Moments of inertia about the body axes

$$J_{xx} = \frac{mr^2}{2} + \frac{mh^2}{6} + \frac{MR^2}{4} + \frac{MH^2}{12} + 2ml^2$$
$$J_{yy} = \frac{mr^2}{2} + \frac{mh^2}{6} + \frac{MR^2}{4} + \frac{MH^2}{12} + 2ml \qquad (8)$$
$$J_{zz} = \frac{MR^2}{2} + 4ml^2$$

The computed MOIs about the body axes are given as:

$$J_{xx} = 0.03\ kgm^2$$
$$J_{yy} = 0.03\ kgm^2$$
$$J_{zz} = 0.04\ kgm^2$$

The cross-products of the moments of inertia are 0 since the quadrotor is assumed to be symmetrical about the $xz-$plane and $yz-$plane.

6. Kinematic Equations

The kinematic equations for the quadrotor are shown in Eqn.(9), and the simplified form assuming small angle approximations is shown in Eqn.(10).

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \cos\theta\sin\phi \\ 0 & -\sin\phi & \cos\theta\cos\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{9}$$

Close to hovering conditions, the small angle approximation is valid; the above matrix is close to the identity matrix, and therefore the angular velocities in the body frame can be seen as angular velocities in the inertial frame.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{10}$$

where p, q, r are the body angular rates. Additionally, ϕ, θ, ψ and $\dot{\phi}, \dot{\theta}, \dot{\psi}$ are the Euler angles and Euler angular rates, respectively.

7. Dynamic Equations

The dynamic equations for the quadrotor are given in Eqn.(11).

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{L}{J_{xx}} \\ \frac{M}{J_{yy}} \\ \frac{N}{J_{zz}} \end{bmatrix} - \begin{bmatrix} \frac{qr}{J_{xx}}\left(J_{zz} - J_{yy}\right) \\ \frac{pr}{J_{yy}}\left(J_{xx} - J_{zz}\right) \\ \frac{pq}{J_{zz}}\left(J_{yy} - J_{xx}\right) \end{bmatrix} \tag{11}$$

where u, v, w and $\dot{u}, \dot{v}, \dot{w}$ are the velocities along the body axes and their derivatives, and p, q, r and $\dot{p}, \dot{q}, \dot{r}$ are the angular rates in the body frame and their derivatives.

8. Final Equations of Motion

The complete 6DOF nonlinear aircraft model for the quadrotor can be summarized as:

Force Equations:

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \frac{1}{m}\begin{bmatrix} F_{total}\sin\phi\sin\psi + F_{total}\cos\phi\sin\theta\cos\psi \\ -F_{total}\sin\phi\cos\psi + F_{total}\cos\phi\sin\theta\sin\psi \\ F_{total}\cos\phi\ cos\theta - mg \end{bmatrix} \tag{12}$$

Moments Equations:

$$\begin{bmatrix} L \\ M \\ N \end{bmatrix} = \begin{bmatrix} (F_{T3} - F_{T4})l \\ (F_{T1} - F_{T2})l \\ (F_{T3} + F_{T4} - F_{T1} - F_{T2})d \end{bmatrix} \tag{13}$$

Dynamics Equations:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{L}{J_{xx}} \\ \frac{M}{J_{yy}} \\ \frac{N}{J_{zz}} \end{bmatrix} - \begin{bmatrix} \frac{qr}{J_{xx}}(J_{zz} - J_{yy}) \\ \frac{pr}{J_{yy}}(J_{xx} - J_{zz}) \\ \frac{pq}{J_{zz}}(J_{yy} - J_{xx}) \end{bmatrix} = \begin{bmatrix} \frac{(F_{T3} - F_{T4})l}{J_{xx}} - \frac{qr}{J_{xx}}(J_{zz} - J_{yy}) \\ \frac{(F_{T1} - F_{T2})l}{J_{yy}} - \frac{pr}{J_{yy}}(J_{xx} - J_{zz}) \\ \frac{(F_{T3} + F_{T4} - F_{T1} - F_{T2})d}{J_{zz}} - \frac{pq}{J_{zz}}(J_{yy} - J_{xx}) \end{bmatrix} \tag{14}$$

Kinematic Equations:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \left(\therefore \textit{small angles approximation} \right) \tag{15}$$

I. LINEARIZED DYNAMICS MODEL

This section describes the linearized dynamic models for use in the controller development.

1. State Vector Representation

The elements of the state vector X are comprised of the components of positions, velocities, Euler angles and angular rates.

$$X = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \phi & \theta & \psi & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T \tag{16}$$

The elements of the control vector U are comprised of the following control inputs.

$$U = \begin{bmatrix} U_z & U_\phi & U_\theta & U_\psi \end{bmatrix}^T \tag{17}$$

where

$$\begin{aligned} U_z &= F_{T1} + F_{T2} + F_{T3} + F_{T4} \\ U_\phi &= \left(F_{T3} - F_{T4} \right) l \\ U_\theta &= \left(F_{T1} - F_{T2} \right) l \\ U_\psi &= \left(F_{T3} + F_{T4} - F_{T1} - F_{T2} \right) d = \left(\tau_3 + \tau_4 - \tau_1 - \tau_2 \right) \end{aligned}$$

where F_{Ti} is the thrust force from i^{th} rotor, and d is the force-to-moment scaling factor.

Taking the derivatives of Eqn.(15) and equate with Eqn.(14) gives Eqn.(18).

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{(F_{T3}-F_{T4})l}{J_{xx}} - \frac{qr}{J_{xx}}(J_{zz}-J_{yy}) \\ \frac{(F_{T1}-F_{T2})l}{J_{yy}} - \frac{pr}{J_{yy}}(J_{xx}-J_{zz}) \\ \frac{(F_{T3}+F_{T4}-F_{T1}-F_{T2})d}{J_{zz}} - \frac{pq}{J_{zz}}(J_{yy}-J_{xx}) \end{bmatrix} \approx \begin{bmatrix} \frac{U_{\phi}}{J_{xx}} - \frac{\dot{\theta}\dot{\psi}}{J_{xx}}(J_{zz}-J_{yy}) \\ \frac{U_{\theta}}{J_{yy}} - \frac{\dot{\phi}\dot{\psi}}{J_{yy}}(J_{xx}-J_{zz}) \\ \frac{U_{\psi}}{J_{zz}} - \frac{\dot{\phi}\dot{\theta}}{J_{zz}}(J_{yy}-J_{xx}) \end{bmatrix} \quad (18)$$

By neglecting the gyroscopic and Coriolis-centripetal effects, the simplified form of Eqn.(18) is shown in Eqn.(19).

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{(F_{T3}-F_{T4})l}{J_{xx}} \\ \frac{(F_{T1}-F_{T2})l}{J_{yy}} \\ \frac{(F_{T3}+F_{T4}-F_{T1}-F_{T2})d}{J_{zz}} \end{bmatrix} = \begin{bmatrix} \frac{U_{\phi}}{J_{xx}} \\ \frac{U_{\theta}}{J_{yy}} \\ \frac{U_{\psi}}{J_{zz}} \end{bmatrix} \quad (19)$$

2. Actuator Dynamics Model

The Qball-X4 uses outrunner brushless motors, and the thrust F_{Ti} produced by the i^{th} rotor is related to the pulse width modulation (PWM) input u_i by the first-order linear transfer function given as:

$$F_{Ti} = K\frac{\omega}{s+\omega}u_i \text{ , for } i = 1,2,3,4 \quad (20)$$

where u_i is the PWM (in percentage of a 20ms duty cycle) input to the i^{th} rotor (i.e., idle throttle occurs when $u = 0.05$ and maximum throttle occurs when $u = 0.10$). Here ω is the motor bandwidth $(\omega = 15\,rad/s)$, and K is a positive gain $(K = 120\,N)$. Although ω and K are theoretically the same

for all four motors, this might not be true in practice. Thus this can be one possible source of modeling errors/uncertainties.

The variable v used to represent the actuator dynamics is given as:

$$v_i = \frac{\omega}{s+\omega} u_i \text{ , for } i = 1,2,3,4 \tag{21}$$

Thus, Eqn.(20) can also be written as:

$$F_i = K v_i \tag{22}$$

3. Roll and Pitch Dynamics Models

Assuming that the rotations about the x and y axes are decoupled, two propellers contribute to the motion in each axis. The thrust generated by each rotor can be calculated from Eqn.(20). The rotation around the center of gravity is produced by the difference in the generated thrusts.

Roll Model:

$$\ddot{\phi} = \frac{(F_{T3} - F_{T4})l}{J_{xx}} = \frac{U_\phi}{J_{xx}} \tag{23}$$

Pitch model:

$$\ddot{\theta} = \frac{(F_{T1} - F_{T2})l}{J_{yy}} = \frac{U_\theta}{J_{yy}} \tag{24}$$

Putting Eqn.(20) through Eqn.(24) into state-space format gives

$$\Delta u_{roll} = u_3 - u_4$$

$$\begin{bmatrix} \dot{\phi} \\ \ddot{\phi} \\ \dot{\nu} \\ \dot{s} \end{bmatrix} = A \begin{bmatrix} \phi \\ \dot{\phi} \\ \nu \\ s \end{bmatrix} + B\Delta u_{roll} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{Kl}{J_{xx}} & 0 \\ 0 & 0 & -\omega & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \dot{\phi} \\ \nu \\ s \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega \\ 0 \end{bmatrix} \Delta u_{roll} \tag{25}$$

$$\Delta u_{pitch} = u_1 - u_2$$

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \\ \dot{\nu} \\ \dot{s} \end{bmatrix} = A \begin{bmatrix} \theta \\ \dot{\theta} \\ \nu \\ s \end{bmatrix} + B\Delta u_{pitch} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{Kl}{J_{yy}} & 0 \\ 0 & 0 & -\omega & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ \nu \\ s \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega \\ 0 \end{bmatrix} \Delta u_{pitch} \tag{26}$$

The fourth state $\dot{s} = \phi$ and $\dot{s} = \theta$ in Eqn.(25) and (26), respectively, are added to the state vector to facilitate the use of an integrator in the feedback structure.

4. Altitude Dynamics Model

The altitude of the quadrotor is affected by all four propellers. The altitude model of the quadrotor can be represented as:

$$\ddot{Z} = \frac{\left(F_{T1} + F_{T2} + F_{T3} + F_{T4}\right)}{m} \cos\phi\cos\theta - g = \frac{U_z}{m} \cos\phi\cos\theta - g \tag{27}$$

where m is the total mass of the quadrotor, and Z is the altitude. Assuming that the rotors produce approximately the same thrust, the altitude dynamics model can be represented in state-space form as shown in Eqn.(28).

$$\begin{bmatrix} \dot{Z} \\ \ddot{Z} \\ \dot{v} \\ \dot{s} \end{bmatrix} = A \begin{bmatrix} Z \\ \dot{Z} \\ v \\ s \end{bmatrix} + Bu = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{4K}{m} & 0 \\ 0 & 0 & -\omega & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} Z \\ \dot{Z} \\ v \\ s \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ -g \\ 0 \\ 0 \end{bmatrix} \tag{28}$$

5. Motion Dynamics Model

The motion of the quadrotor along the horizontal plane of the *ENU* frame can be represented by Eqn.(29)

$$\begin{bmatrix} \ddot{X} \\ \ddot{Y} \end{bmatrix} = \begin{bmatrix} a_x \\ a_y \end{bmatrix} = \begin{bmatrix} \frac{U_z}{m}\left(\sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi\right) \\ \frac{U_z}{m}\left(-\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi\right) \end{bmatrix} \tag{29}$$

With small angle approximation, and assuming the rotors produce approximately the same thrust, the motion dynamics of the quadrotor can be represented as:

$$\begin{bmatrix} \dot{X} \\ \ddot{X} \\ \dot{v} \\ \dot{s} \end{bmatrix} = A \begin{bmatrix} X \\ \dot{X} \\ v \\ s \end{bmatrix} + B\mathrm{u} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{4K}{m}\theta & 0 \\ 0 & 0 & -\omega & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} X \\ \dot{X} \\ v \\ s \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega \\ 0 \end{bmatrix} u$$

$$\begin{bmatrix} \dot{Y} \\ \ddot{Y} \\ \dot{v} \\ \dot{s} \end{bmatrix} = A \begin{bmatrix} Y \\ \dot{Y} \\ v \\ s \end{bmatrix} + B\mathrm{u} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{4K}{m}\phi & 0 \\ 0 & 0 & -\omega & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} Y \\ \dot{Y} \\ v \\ s \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega \\ 0 \end{bmatrix} u \tag{30}$$

6. Yaw Dynamics Model

The relationship between the torque τ_i generated by the $i^{th}\ (i = 1, 2, 3, 4)$ rotor and the PWM input to each rotor u_i can be represented by Eqn.(31).

$$\tau_i = K_{yaw} u_i \tag{31}$$

where K_{yaw} is a positive gain $\left(K_{yaw} = 4\ Nm\right)$.

The motion in the yaw axis is caused by the difference between the torque exerted by the two clockwise and two counter-clockwise propellers. The yaw dynamics can be written as:

$$\begin{aligned} \Delta u &= u_3 + u_4 - u_1 - u_2 \\ \ddot{\psi} &= \frac{\left(F_{T3} + F_{T4} - F_{T1} - F_{T2}\right) d}{J_{zz}} = \frac{K_{yaw}}{J_{zz}} \Delta u \end{aligned} \tag{32}$$

In state-space representation can be written as:

$$\begin{bmatrix} \dot{\psi} \\ \ddot{\psi} \end{bmatrix} = A \begin{bmatrix} \psi \\ \dot{\psi} \end{bmatrix} + B\Delta u = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \psi \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{K_{yaw}}{J_{zz}} \end{bmatrix} \Delta u \tag{33}$$

7. Control Mixer

Control mixing combines the outputs from the altitude, roll, pitch and heading control channels to generate the following command inputs to the rotors:

$$\begin{aligned} u_1 &= \nu_{th} + \nu_\theta + \nu_\psi \\ u_2 &= \nu_{th} - \nu_\theta + \nu_\psi \\ u_3 &= \nu_{th} + \nu_\phi - \nu_\psi \\ u_4 &= \nu_{th} - \nu_\phi - \nu_\psi \end{aligned} \tag{34}$$

where ν_{th}, ν_ϕ, ν_θ and ν_ψ are the output commands in PWM from the altitude, roll, pitch and heading control channels, respectively. The input commands to the individually controlled rotor are u_1, u_2, u_3 and u_4.

J. SUMMARY OF SYSTEM PARAMETERS

The following system parameters shown in Table 4 were obtained from experimental results (Quanser 2011).

Table 4: System Parameters.

Parameter	Symbol	Value
Thrust coefficient	K	120 N
Actuator Bandwidth	ω	15 rad/s
MOI about x-axis	J_{xx}	0.03 kgm^2
MOI about y-axis	J_{yy}	0.03 kgm^2
MOI about z-axis	J_{zz}	0.04 kgm^2
Total mass	m	1.4 kg
Torque coefficient	K_{yaw}	4 Nm
Length of moment arm	l	0.2 m

IV. SIMULINK IMPLEMENTATION

A. OVERVIEW

The implementation of the 6DOF simulation model in Simulink is described in this chapter. The default controller provided by Quanser with the Qball-X4 is illustrated. A new PID controller developed for each of the control channels for basic navigation is also proposed.

B. OVERVIEW OF 6DOF SIMULATION MODEL

An overview of the 6DOF simulation model is shown in Figure 15. It is comprised of five main modules and four auxiliary ones.

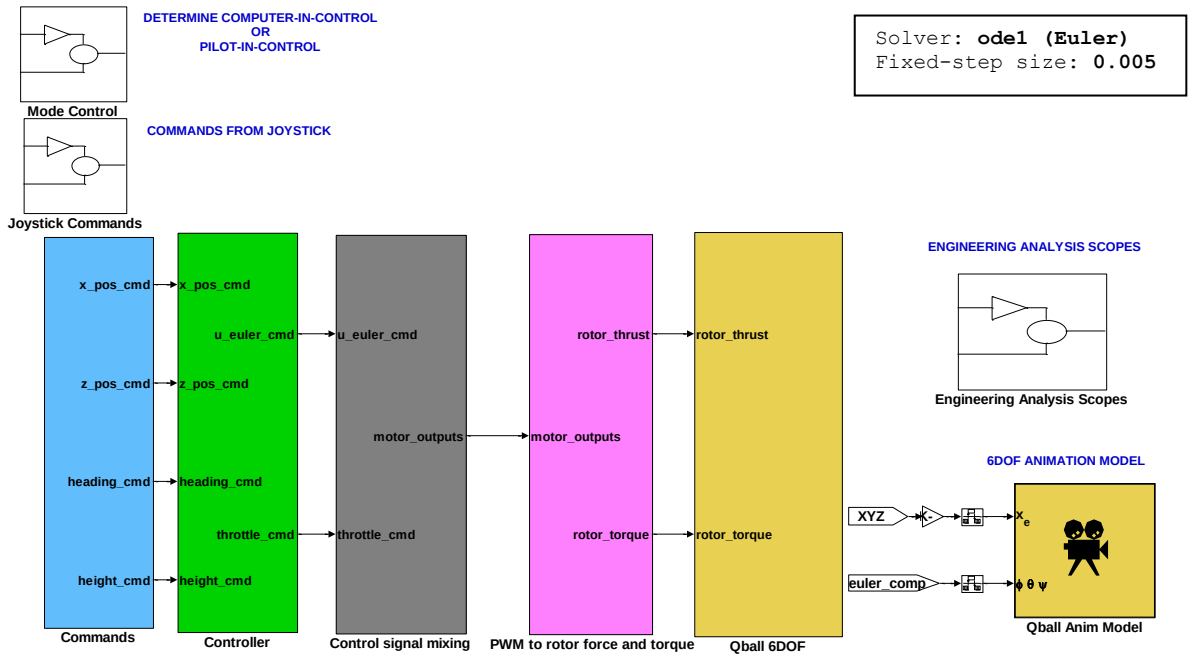


Figure 15: Overview of 6-DOF Simulation Model.

The main modules are 1) Commands, 2) Controller, 3) Control Signal Mixing, 4) PWM to Rotor Force and Torque and 5) Qball 6DOF subsystem blocks.

The auxiliary modules are 1) Mode Control, 2) Joystick Commands, 3) Engineering Analysis Scopes and 4) Qball Animation Model blocks.

The configuration parameters setting for the simulation model was set to be similar to the actual Qball-X4 controller model, which employs ODE1 (Euler) for the solver with a step size of 0.005 sec (200 Hz).

A description of each module is provided in Table 5.

Table 5: Modules and Their Descriptions.

Block Name	Function
Main Modules	
Commands	Consists of a waypoint management module which provides the high-level commands to the controller module.
Controller	Consists of four control channels, namely X position, Z position, Height and Heading commands. Outputs commands from each channel are in PWM.
Control signal mixing	Combines the commands from each control channel and outputs PWM commands to individual motor.
PWM to rotor force and torque	Maps PWM inputs to corresponding force and torque generated by each motor.
Qball 6DOF	Consists of the equations of motion

	of the Qball-X4 quadrotor to provide the instantaneous states of the aircraft.

Auxiliary Modules	
Mode Control	User-defined mode: Computer-In-Control (CIC) or Pilot-In-Control (PIC).
Joystick Commands	Receives commands from manual joystick.
Engineering Analysis Scopes	Consists of various scopes to compare feedbacks to commands for engineering analysis purposes during simulation run.
6DOF Animation Model	Provides 3D animation of the quadrotor during simulation run.

C. COMMANDS MODULE

The waypoint management state machine resides in the Commands module. It handles the waypoint updates for the quadrotor and reports on the current state of the aircraft. The waypoint management state machine module is shown in Figure 16.

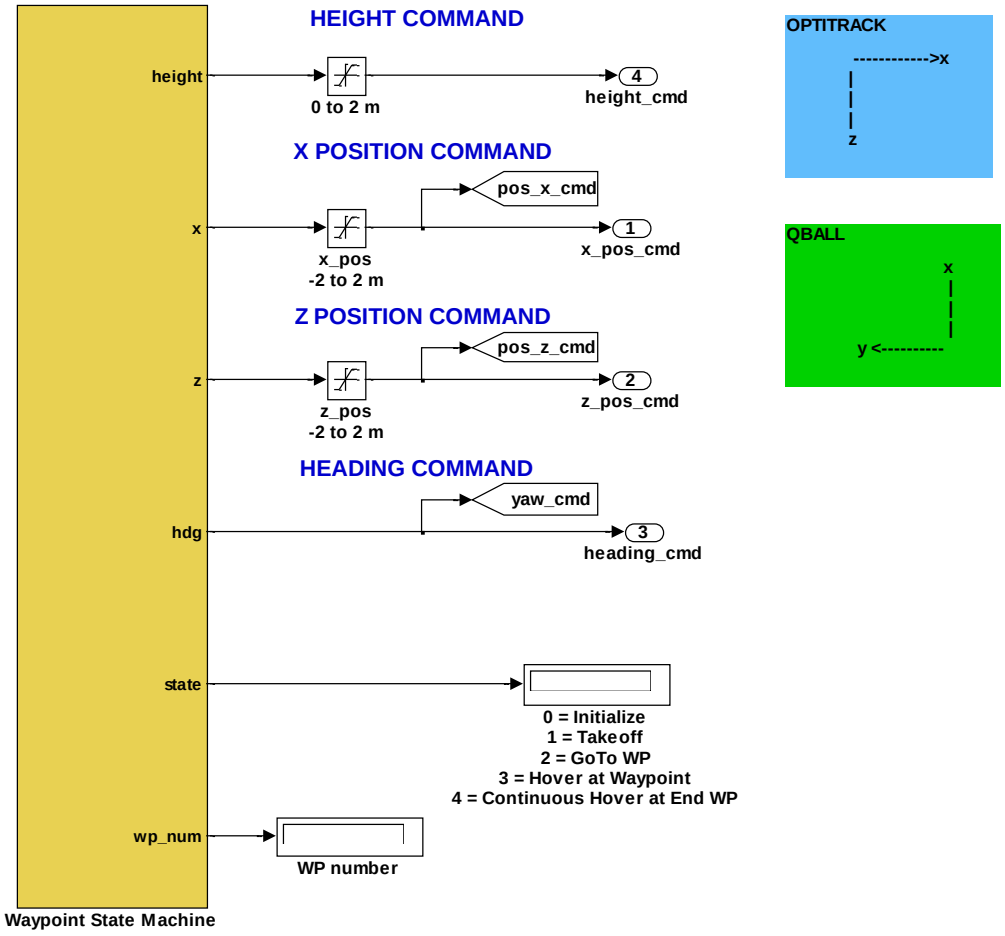


Figure 16: Waypoint Management State Machine Block.

The outputs from the Waypoint Management State Machine are the position commands, altitude command, heading command, the current flight state of the quadrotor and the waypoint number.

The process logic within the state machine is described in Figure 17.

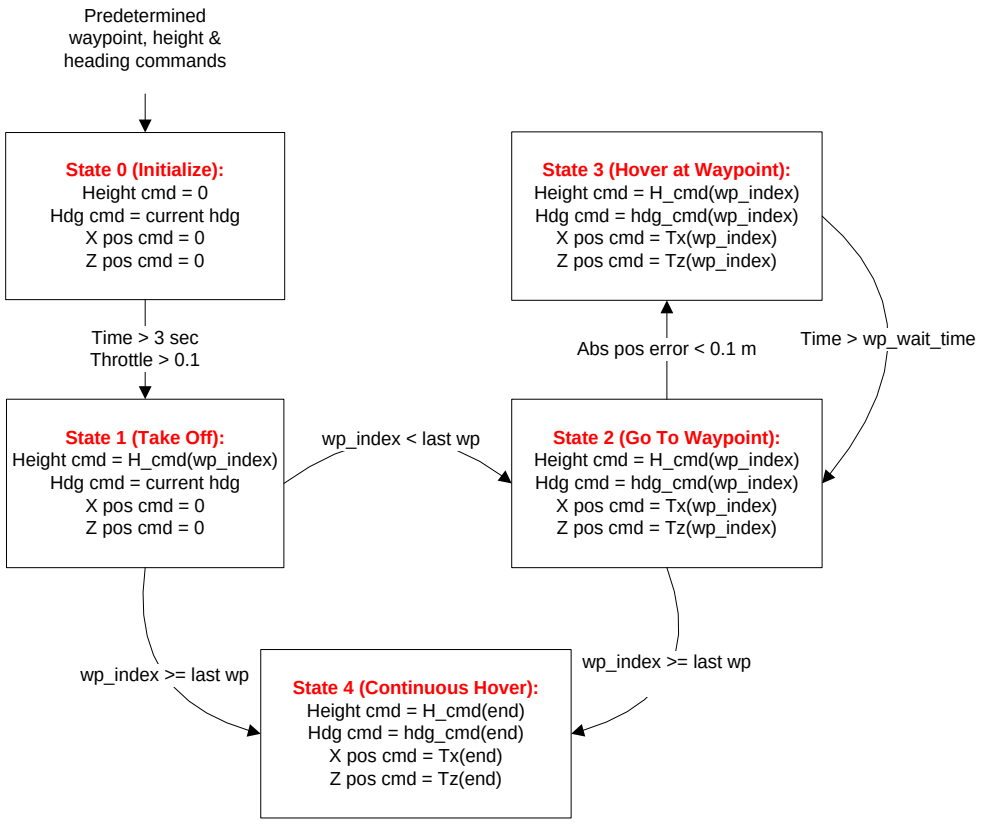


Figure 17: Process Logic in Waypoint Management State Machine.

D. DEFAULT CONTROLLER DESIGN

There are four decoupled control channels residing in the controller module, namely the Position outer-loop control, Attitude (Pitch and Yaw) inner-loop control, Heading control and Altitude control. The position and attitude controllers have very similar configurations since the quadrotor is symmetrical about the $xz-$plane and $yz-$plane.

1. Position Feedback Control

The schematic diagram of the default position controller developed by Quanser is illustrated in Figure

18, with the actual implementation shown in Figure 19 and Figure 20.

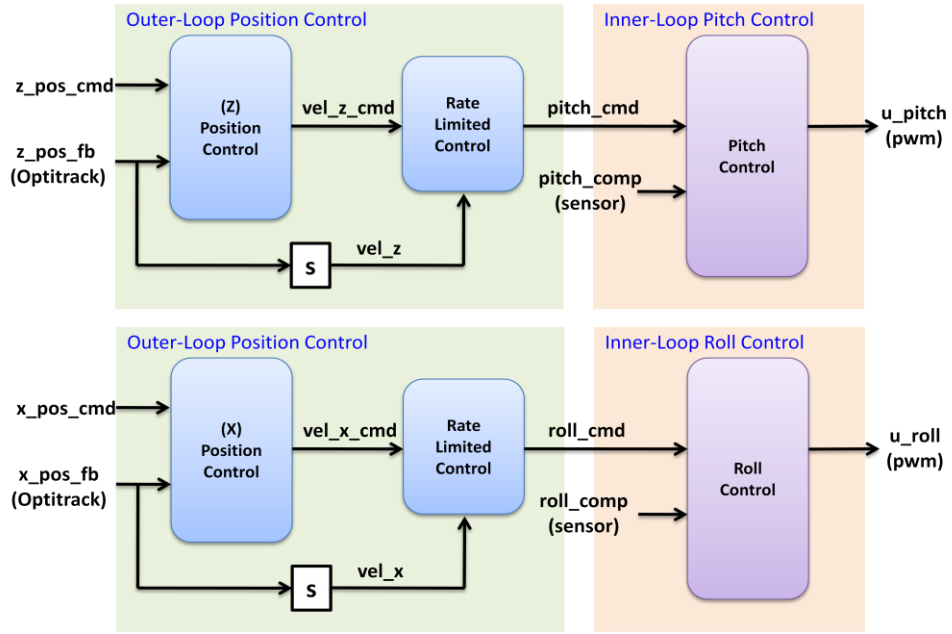


Figure 18: Schematic Diagram of the Default Position Controllers.

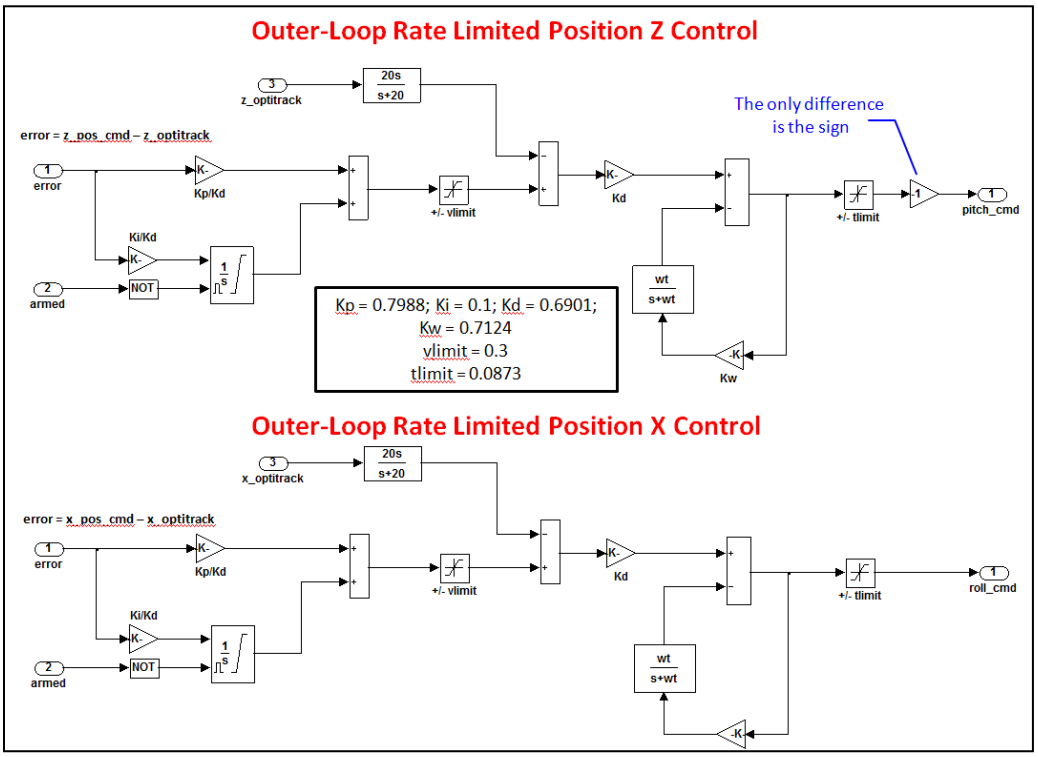


Figure 19: Actual Implementation of Outer Loop Position Control.

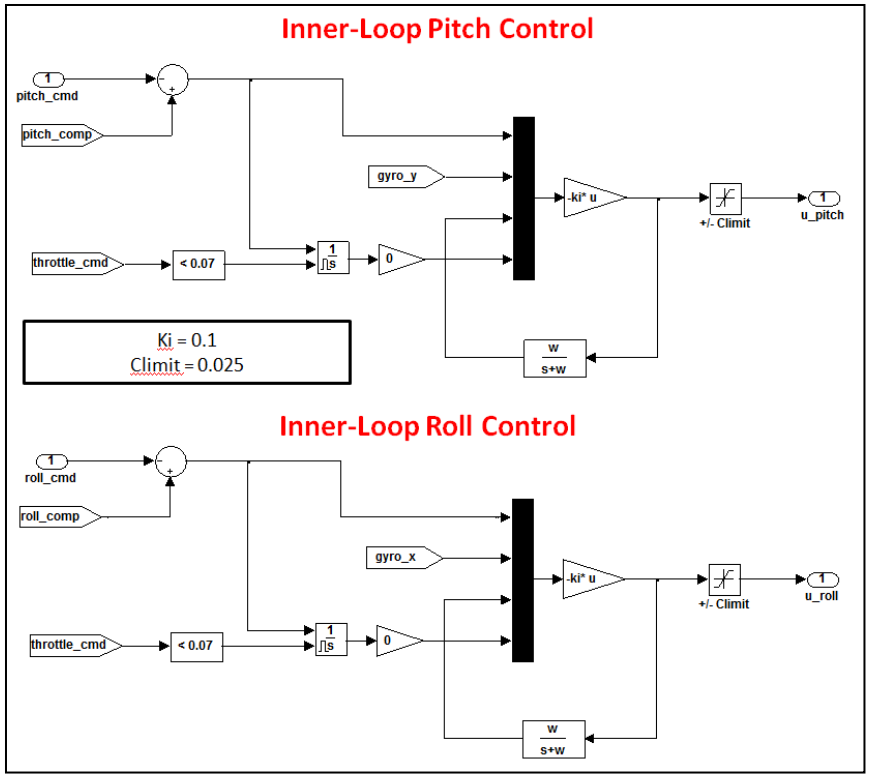


Figure 20: Actual Implementation of Inner Loop Pitch and Roll Control.

Velocities are estimated from the derivative of positions. The position information is obtained from the Optitrack system, while the roll and pitch attitude are computed from the inertial measurement unit (IMU) sensor onboard the Qball-X4.

2. Heading Feedback Control

The heading controller, which incorporates a yaw damper, adopted a simpler architecture as shown in Figure 21. Heading control is performed as a separate process, independent of the pitch and roll of the quadrotor. Figure 22 displays the actual implementation of the heading controller in Simulink.

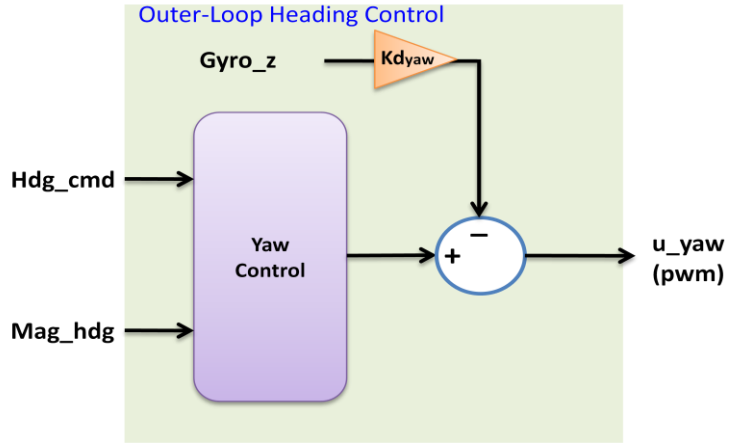


Figure 21: Schematic Diagram of the Default Heading Controller.

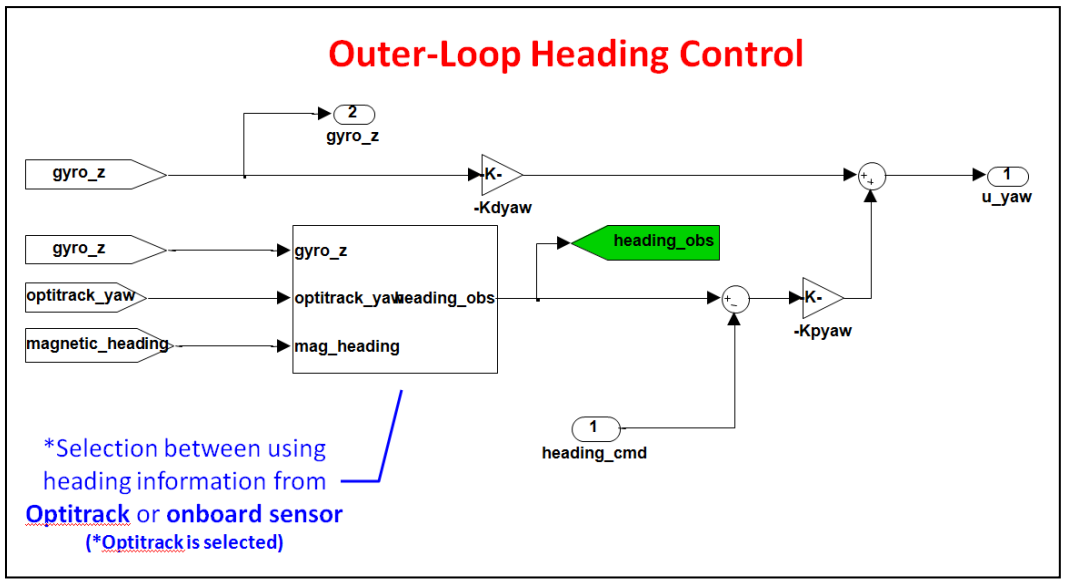


Figure 22: Actual Implementation of Outer Loop Heading Controller.

3. Altitude Feedback Control

The schematic diagram of the altitude controller is given in Figure 23. It has a sigmoid modifier block which alters the altitude command so that it has a sigmoid profile instead of a step profile. The sonar sensor at the base of the quadrotor provides the altitude feedback. Gain

scheduling was also implemented so that it uses a different set of integrator gains during landing and takeoff. Figure 24 shows the actual implementation of the altitude controller in Simulink.

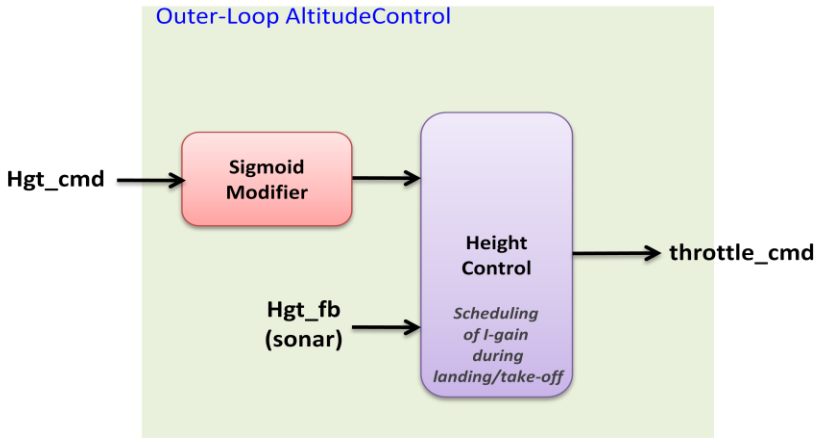


Figure 23: Schematic Diagram of the Default Altitude Controller.

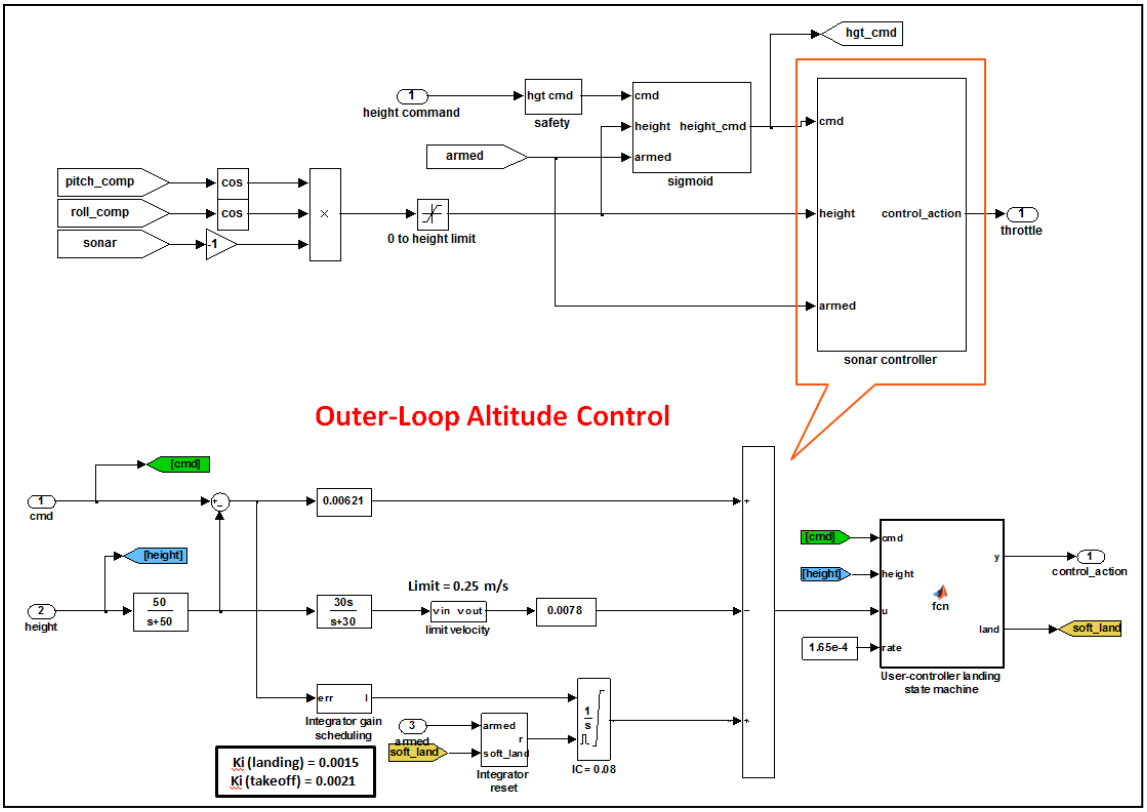


Figure 24: Actual Implementation of Outer Loop Altitude Controller.

E. PID CONTROLLER DESIGN

It was realized that the default controller provides only basic control to the quadrotor. To improve the flight performance, a new PID controller for each of the control channels was proposed. Its main advantages include having a simple structure, as well as ease of use and tuning.

1. Proposed PID Controllers

The proposed control architecture (Figure 25) consists of a Position-to-Velocity and Velocity-to-Roll outer-loop PID controller and an inner-loop Roll PID controller for the roll control channel. Similarly for the pitch control channel, the proposed architecture consists of a Position-

to-Velocity and Velocity-to-Pitch outer-loop PID controller and an inner-loop Pitch PID controller.

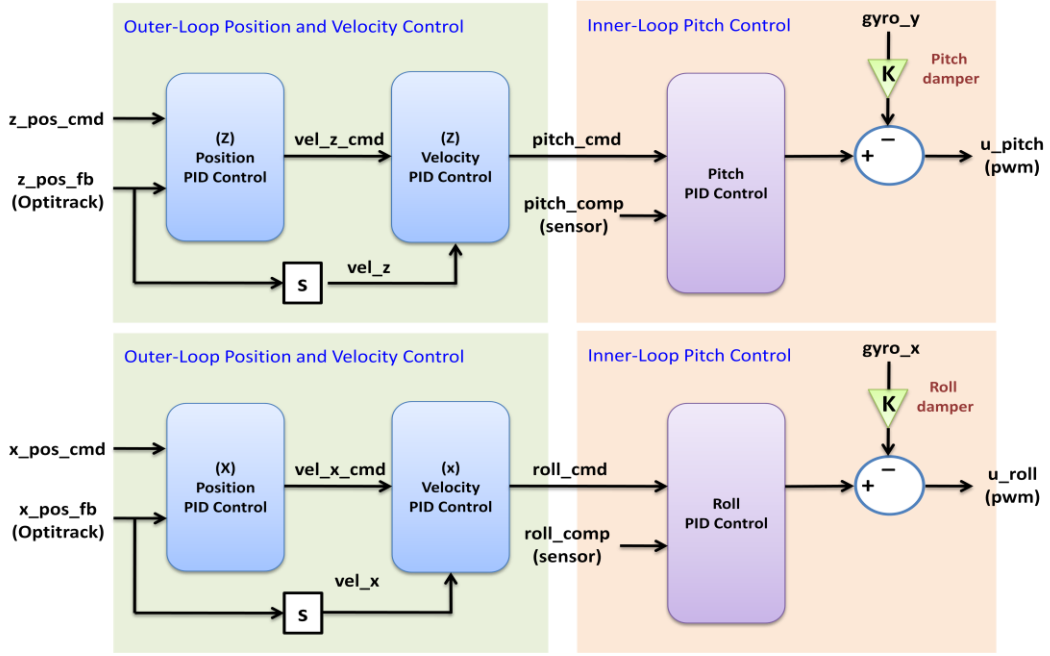


Figure 25: Proposed Roll and Pitch PID Controller.

The actual implementation of the outer-loop Position-to-Velocity and Velocity-to-Pitch/Roll PID controllers is shown in Figure 26, and the inner-loop pitch/roll PID controllers are shown in Figure 27.

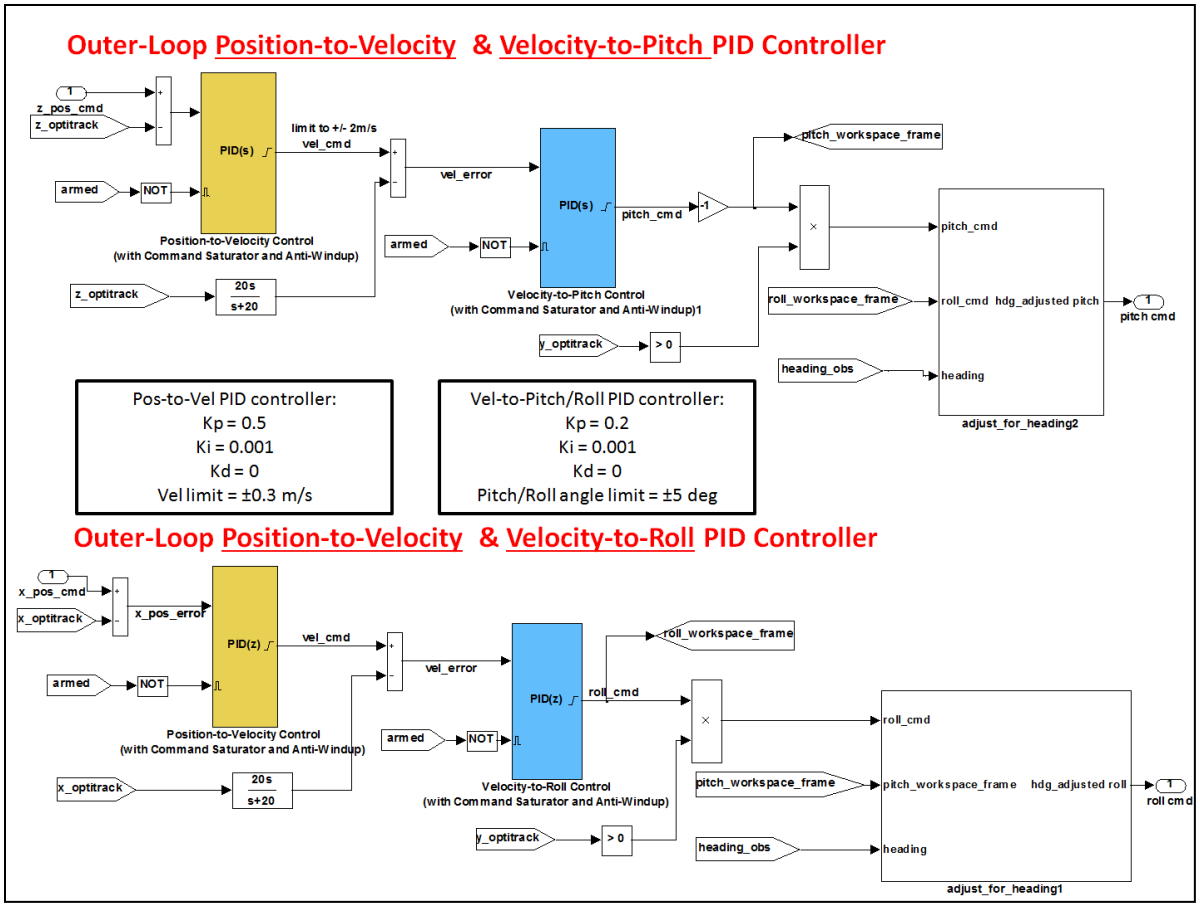


Figure 26: Outer Loop Position-to-Velocity and Velocity-to-Roll/Pitch PID Controller.

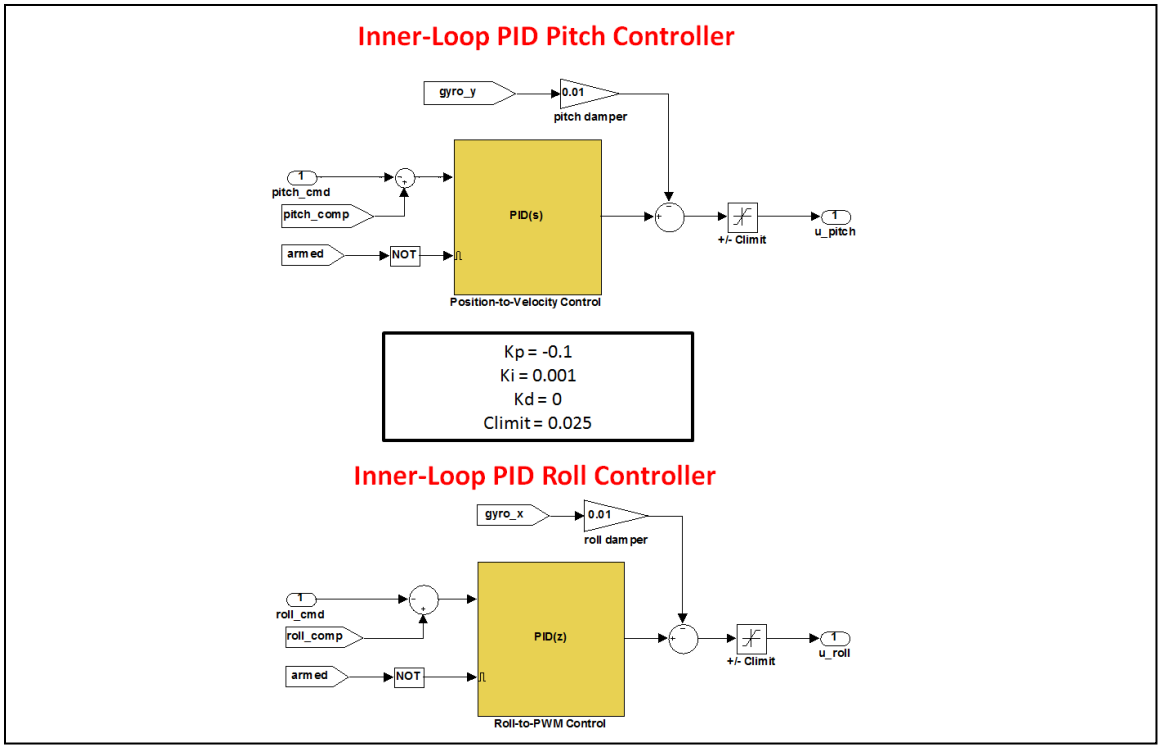


Figure 27: Inner Loop Pitch and Roll PID Controller.

2. Heading Feedback Control

No change in architecture was done to the heading feedback control. Since experimental results show the performance of the heading controller to be sufficient for the work of this thesis, such changes were unnecessary.

3. Altitude Feedback Control

The default altitude controller was used; however, the gains were retuned, since the original set of gains exhibited poor altitude control performance.

F. CONTROL SIGNAL MIXING

The main purpose of the control signal mixer is to merge the command outputs from the individual control channels in order to achieve the control objectives. The output from the control signal mixer consists of the PWM input commands to the individual motors. The configuration of the control signal mixing block is shown in Figure 28.

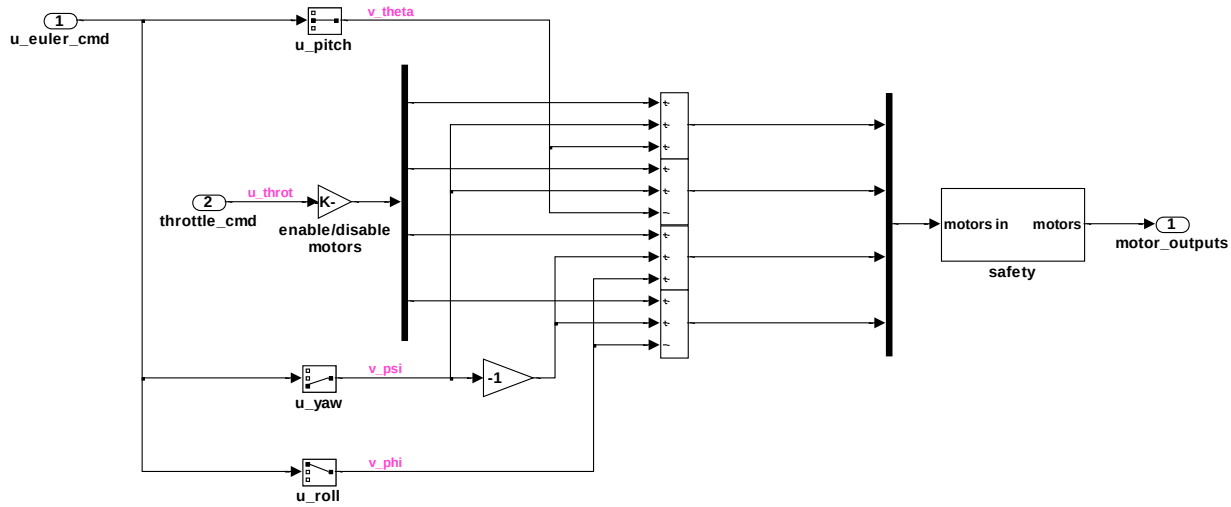


Figure 28: Control Signal Mixing Module.

G. PWM TO ROTOR FORCE AND TORQUE

This block typically converts PWM into the corresponding rotor forces and torques using the relationships shown in Eqn.(20) and Eqn.(31). The schematics for the PWM to rotor force and torque module are shown in Figure 29. The saturation limits for the PWM are 0.05 and 0.1, which correspond to 1 ms and 2 ms, respectively.

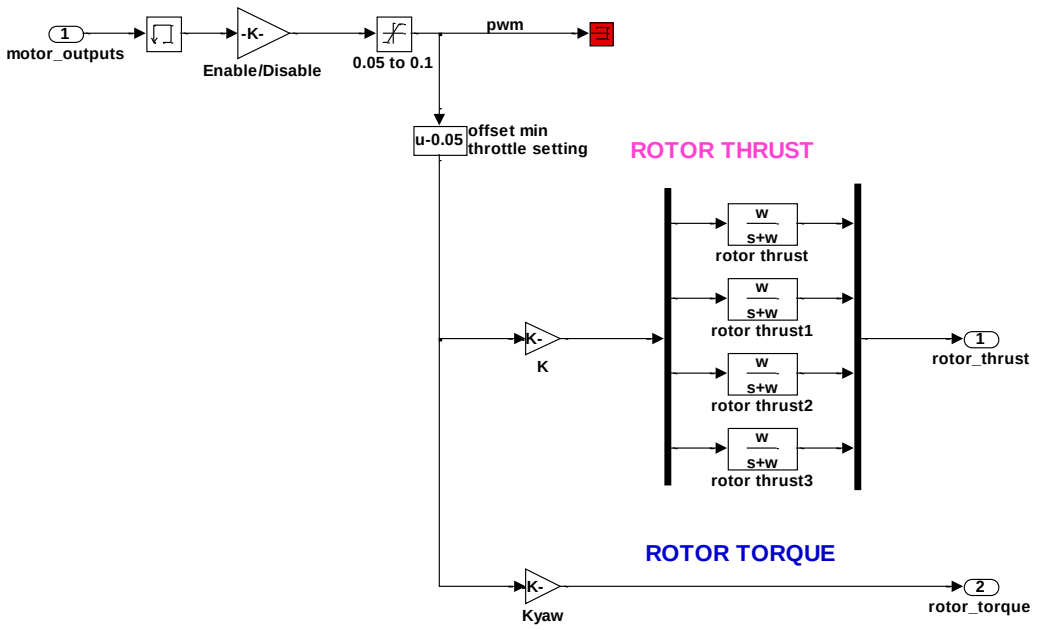


Figure 29: PWM to Rotor Force and Torque Module.

H. QBALL-X4 6DOF MODEL

This block computes the states of the quadrotor in real-time using the equations of motion derived earlier in Chapter III. It should be noted that in reality position information is obtained via the external Optitrack motion capture system. In simulation, however, the position and orientation are computed using the force and moment equations. Figure 30 shows the schematic diagram of the Qball-X4 6DOF module. The red box as indicated in Figure 30 creates an imaginary ground so that the quadrotor would not descend below ground level.

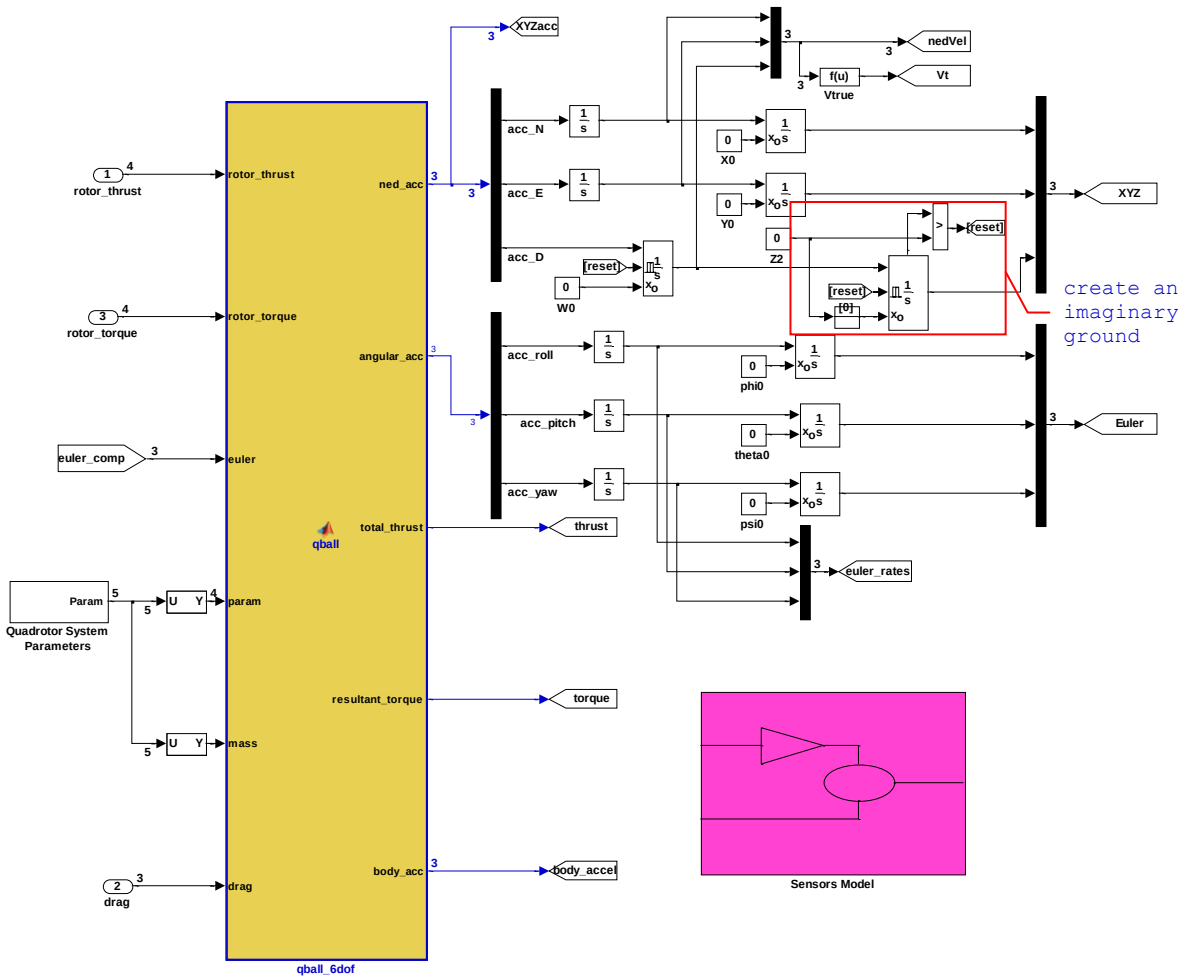


Figure 30: Qball-X4 6DOF Block.

I. QBALL-X4 ANIMATION MODEL

This block creates a 3D animation of the simulation results in real-time. The inputs to the block are the Euler angles (ϕ, θ, ψ) and position information (x, y, z). The advantages of having an animation include acceleration and simplification of error analysis and self-explanatory 3D animation of the vehicle behavior. A snapshot of the 3D animation of the quadrotor performing a square trajectory flight profile is shown in Figure 31.

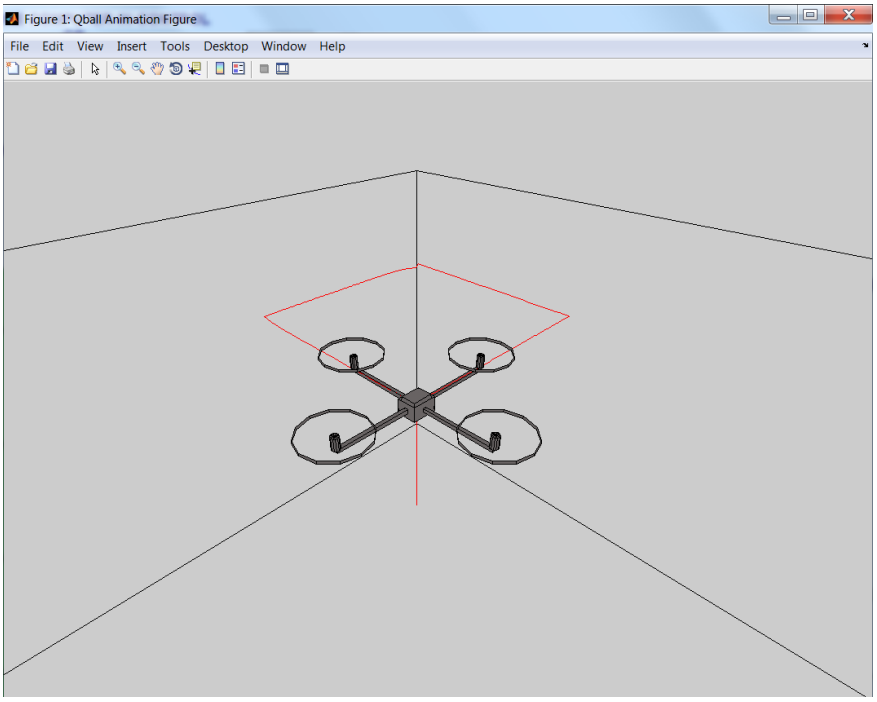


Figure 31: 3D Animation of Quadrotor Performing a Square Trajectory Flight Profile.

THIS PAGE INTENTIONALLY LEFT BLANK

V. DILUTION OF PRECISION

A. INTRODUCTION

Dilution of precision (DOP) is typically used in global positioning system (GPS) and geomatics engineering to specify the additional multiplicative effect of GPS geometry on GPS precision. For this thesis, this concept is used to examine the Optitrack motion capture system's DOP and how it affects the accuracy in which the system can determine position. DOP comes in various flavors, including geometrical (GDOP), positional (PDOP), horizontal (HDOP), vertical (VDOP) and time (TDOP).

B. GEOMETRY

The idea of GDOP is to examine how errors in the measurement affect the final estimation of the state, such that

$$GDOP = \frac{\Delta Output\ Location}{\Delta Measurements} \tag{35}$$

It is desired that small errors in the measurement will not lead to significant changes in the output location, since a large change indicates that the solution is highly sensitive to errors.

Examples of acceptable and poor GDOP resulting from the geometry of the location system are shown in Figure 32. When the visible localization cameras are close together, the geometry is said to be weak, and the DOP value is high. When the cameras are far apart, the geometry is said to be strong, and the DOP value is small.

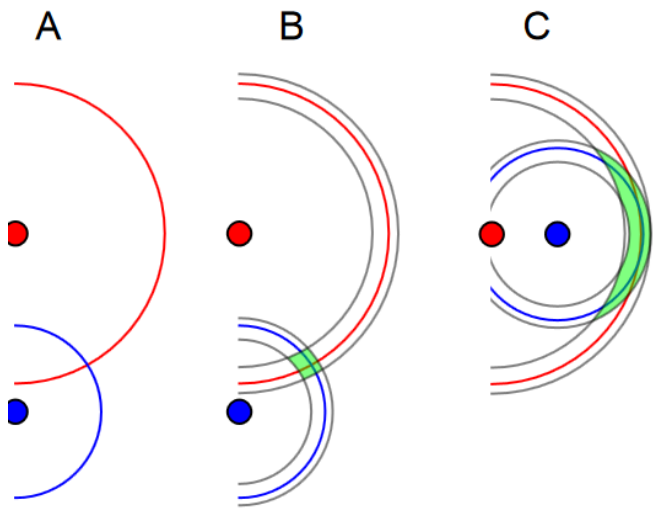


Figure 32: Geometric Dilution of Precision (A) Triangulation (B) Triangulation with error (C) Triangulation with error and poor GDOP (From Xoneca 2013).

C. PSEUDORANGE MEASUREMENTS

The Optitrack system computes the vehicle's three-dimensional coordinates from three or more simultaneous pseudorange measurements. The range can be measured from the infrared (IR) light emitted from the cameras as it reflects back to the camera from the reflective markers attached to the vehicle. The basic pseudorange model can be given by

$$P_i = \rho_i + c\left(dT_{2i} - dT_{1i}\right) + e \tag{36}$$

where P_i is the pseudorange, ρ_i is the geometric range between the i^{th} camera and the quadrotor, c is the speed of light $\left(3.0\times10^{8}\ \mathrm{ms}^{-1}\right)$, dT_{2i} and dT_{1i} are the time biases in the camera system at emission and receiving of the IR pulse. The measurement noise is accounted for by e. There are n

such equations to solve using the n simultaneous measurements.

Without loss of generality, Eqn.(36) can be reduced to the form shown in Eqn.(37).

$$P_i = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} + E_i \tag{37}$$

To determine the quadrotor coordinates, the pseudorange equations are first linearized using some initial estimates for the vehicle position (the linearization point).

$$\delta P_c = H \delta x \tag{38}$$

where δP_c is the n-length vector of differences between the corrected pseudorange measurements and the modeled pseudorange values based on the linearization point. The vector of corrections to an unknown position is δx. In Eqn.(39), H is the $n \times 3$ matrix of partial derivatives.

$$H = \begin{bmatrix} \frac{(x - x_1)^2}{\rho_1} & \frac{(y - y_1)^2}{\rho_1} & \frac{(z - z_1)^2}{\rho_1} \\ \frac{(x - x_2)^2}{\rho_2} & \frac{(y - y_2)^2}{\rho_2} & \frac{(z - z_2)^2}{\rho_2} \\ \vdots & \vdots & \vdots \\ \frac{(x - x_n)^2}{\rho_n} & \frac{(y - y_n)^2}{\rho_n} & \frac{(z - z_n)^2}{\rho_n} \end{bmatrix} \tag{39}$$

Eqn.(38) is solved using the maximum likelihood parameter estimation method, which gives the following solution form

$$\delta x = -\left(H^T W H\right)^{-1} H^T W \delta P_c \tag{40}$$

where $W = \sigma_0^2 C_{\Delta P_c}^{-1}$ is the weight matrix, which is characterized by the differences in the errors of the simultaneous measurements. The inverse term in the weight matrix is the covariance matrix of the pseudorange errors, and σ_0^2 is a scale factor (priori variance of unit weight).

D. COVARIANCE MATRIX

The covariance law determines how the estimated parameters obtained from Eqn.(40) are affected by the pseudorange measurements and model errors.

$$C_{\Delta x} = \left[\left(H^T W H\right)^{-1} H^T W\right] C_{\Delta P_c} \left[\left(H^T W H\right)^{-1} H^T W\right]^T = \left(H^T C_{\Delta P_c}^{-1} H\right)^{-1} \qquad (41)$$

where $C_{\Delta x}$ is the covariance matrix of the parameter estimates. If it is assumed that the measurement and model errors are the same for all observations with a standard deviation σ and they are uncorrelated, then $C_{\Delta P_c} = I\sigma^2$. Eqn.(41) can be simplified to that of the least-squares parameter estimation solution

$$C_{\Delta x} = \left(H^T H\right)^{-1} \sigma^2 = G^{-1}\sigma^2 \qquad (42)$$

If we further assume that the measurement errors and model errors are independent, then the standard deviation σ is obtained via the root-mean-square of these errors.

E. DILUTION OF PRECISION

The geometric dilution of precision (GDOP) measures the overall quality of the least-squares solution and is defined as

$$\sigma_G = \sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2 + \sigma_t^2} = \sqrt{trace\left(G^{-1}\right)} \qquad (43)$$

where $\sigma_x^2, \sigma_y^2, \sigma_z^2$ are the variances in position estimates; σ_t^2 is the variance in time offset estimate, which is zero in the case for the Optitrack since it is using the same clock.

The quality of the specific three-dimensional position component estimates can be given as

$$\begin{aligned} &position: && \sigma_P = \sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2} \\ &horizontal: && \sigma_H = \sqrt{\sigma_x^2 + \sigma_y^2} \\ &vertical: && \sigma_V = \sqrt{\sigma_z^2} \end{aligned} \tag{44}$$

The corresponding position, horizontal and vertical DOPs can be computed using

$$\begin{aligned} PDOP &= \frac{\sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2}}{\sigma} \\ HDOP &= \frac{\sqrt{\sigma_x^2 + \sigma_y^2}}{\sigma} \\ VDOP &= \frac{\sigma_z}{\sigma} \end{aligned} \tag{45}$$

VDOP values are generally larger than HDOP values because all the cameras are above the vehicle.

F. TEST SETUP AND RESULTS

Figure 33 shows a procedure that was used to collect the position measurement errors from the origin of the coordinate frame used by the Qball-X4 in the laboratory setup (see Appendix A for laboratory setup). The Qball-X4 was placed at the point of origin and then moved in 1 cm increments along the z-axis.

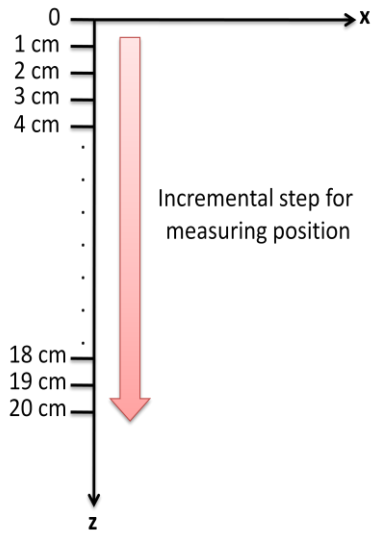


Figure 33: Procedure for Determining Position Accuracy of the Optitrack System.

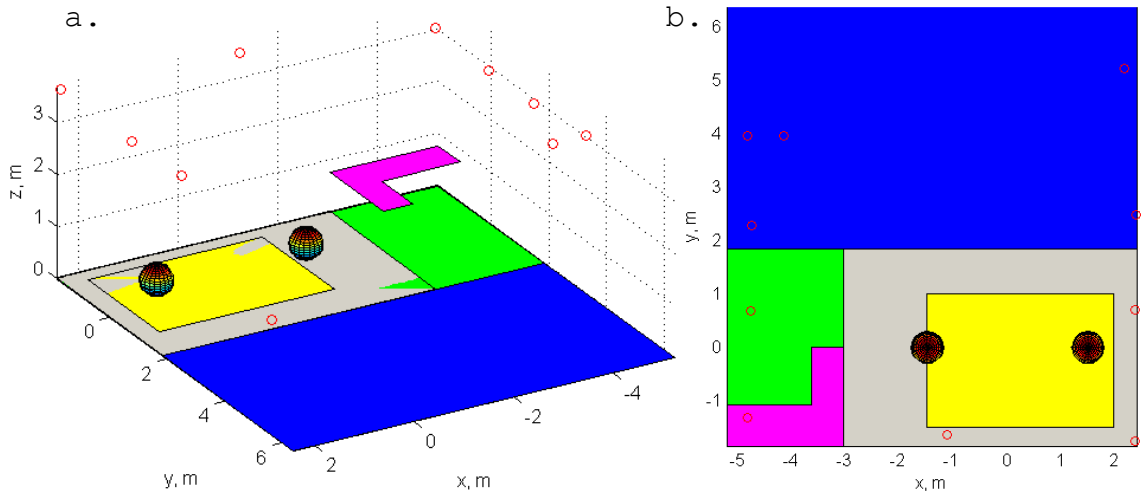


Figure 34: 3D projection of the ASEIL setup (a), and its bird-eye's view (b).

Figure 34(a) shows the three-dimensional projection of the space dedicated for the ASEIL lab with two Quanser Qballs sitting on the floor, and Figure 34(b) represents the bird's-eye view of the lab. Circles (red) on both figures indicate locations of the ten V100:R2 low-end quality

Optitrack cameras. The physical coordinates of the camera locations are given in Table 6.

Table 6: Location of the Optitrack Cameras in ASEIL Lab.

Camera	x coordinate, m	y coordinate, m	z coordinate, m
1	-4.853995	5.619800	3.395093
2	-4.173877	3.953969	3.383460
3	-4.773333	2.288210	3.340947
4	-4.853995	-1.311777	3.318943
5	-4.798242	0.6756070	3.327053
6	-1.127673	-1.624753	3.596775
7	2.396973	-1.749930	3.660483
8	2.3911163	0.7045230	3.656286
9	2.410183	2.483713	3.707655
10	2.184218	5.220389	3.700000

There is no doubt that the correct number and placement of the Optitrack cameras is of fundamental importance to successful tracking of moving objects. Table 7 lists the recommended number of cameras for various room sizes along with approximate workspace volumes that are recommended for the various camera packages.

Table 7: Setup Size and Capture Volume for Various Camera Packages.

Number of Cameras	Setup volume (room size) L×W×H	Workspace volume (experiment) L×W×H
6	4m × 4m × 3m	1.5m × 1.5m × 1m
12	6m × 6m × 3m	3m × 3m × 1m
18	7m × 7m × 3m	3.5m × 3.5m × 1.5m
24	7m × 7m × 3m	4m × 4m × 1.5m

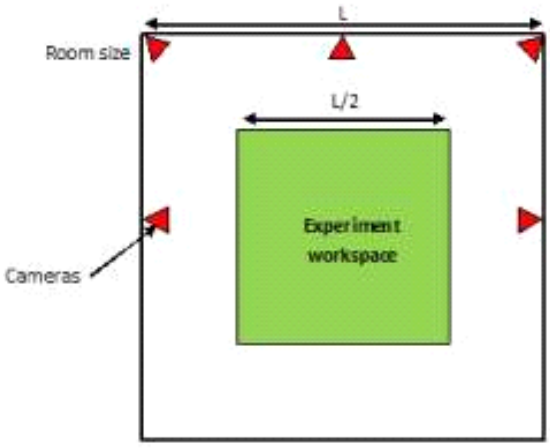


Figure 35: Example of the Camera Setup Inside a Room as Viewed from Above.

The general rules for camera placement (since no single "perfect" setup exists) include the following:

- Cameras should be mounted in as large a perimeter as possible (typically not larger than 7 m across).
- The camera fields of view (FOV) must overlap so that objects are trackable in the workspace; so mounting them farther away allows for a larger overlapping volume.
- Cameras should be mounted higher up along the walls or ceiling to provide an optimal viewpoint and create a large overlapping volume.

As shown in Table 7, even though the ASEIL space is not very big, it definitely requires at least 18 to 24 cameras (as compared to just 10 currently available in the ASEIL lab). Out of the $8.6\,m \times 8.0\,m$ area, only $3.9\,m \times 5.8\,m$ is dedicated for an experimental fly zone with the floor covered by a non-reflective material. Adding a safety buffer further shrinks the flyable zone to about $2.6\,m \times 3.6\,m$

area. The Optitrack cameras are distributed over the perimeter of the entire room, and the actual flyable zone is situated at the corner of the room. Another disadvantage is the low ceiling such that the highest mounted cameras are at 3.7 m above the floor.

Figure 36 shows the isolines of DOP at four different altitude levels: at 0.5m, 1.2m, 1.8 m and 2.5 m above the floor, which is an indication of the "pureness" of the ASEIL setup. Obviously, with the Qball-X4 flying closer in the plane containing the Optitrack cameras, the DOP degrades and becomes quite nonlinear at the corners.

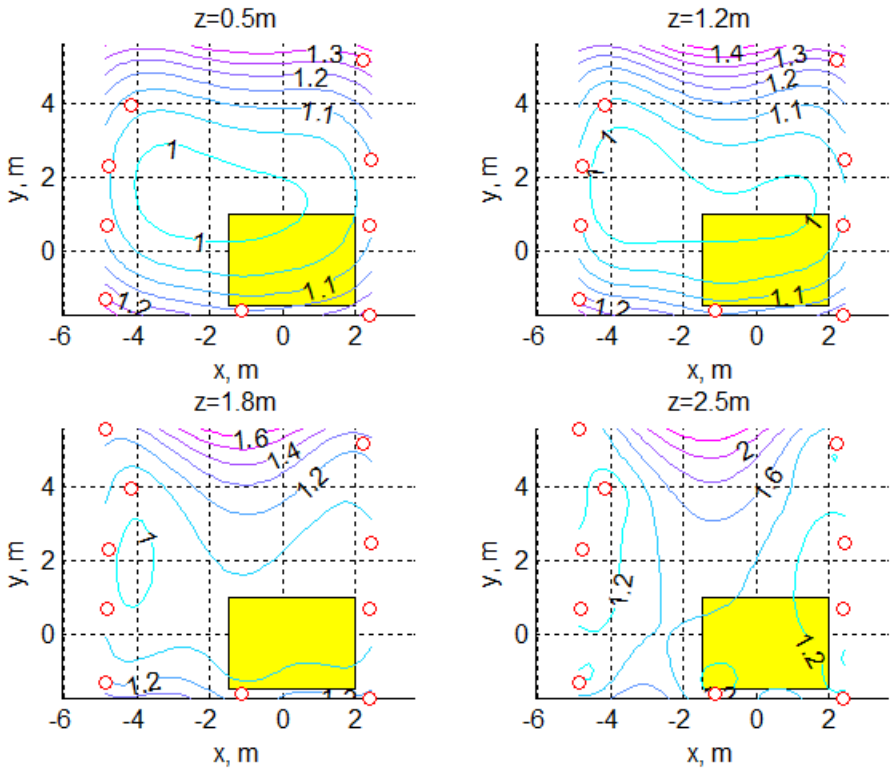


Figure 36: Isolines of DOP for a 10-camera ASEIL Setup at 0.5m, 1.2m, 1.8m and 2.5m Altitude.

For the sake of comparison, Figure 37 shows what would happen if there were more height available. If the ceiling

was one or two meters higher, the DOP within the flyable zone would be much better.

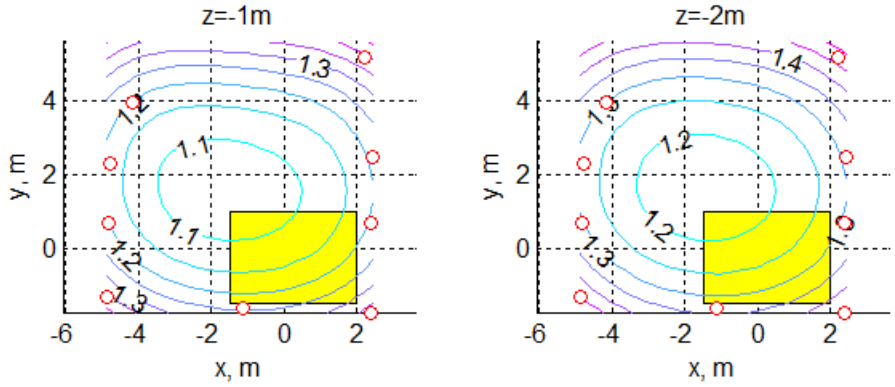


Figure 37: Isolines of DOP for a 10-camera ASEIL Setup at -1m and -2m Altitude.

Figure 38 shows the case if we have two cameras installed at each existing camera location in the ASEIL lab, totaling of 20 cameras. It is expected that DOP would improve.

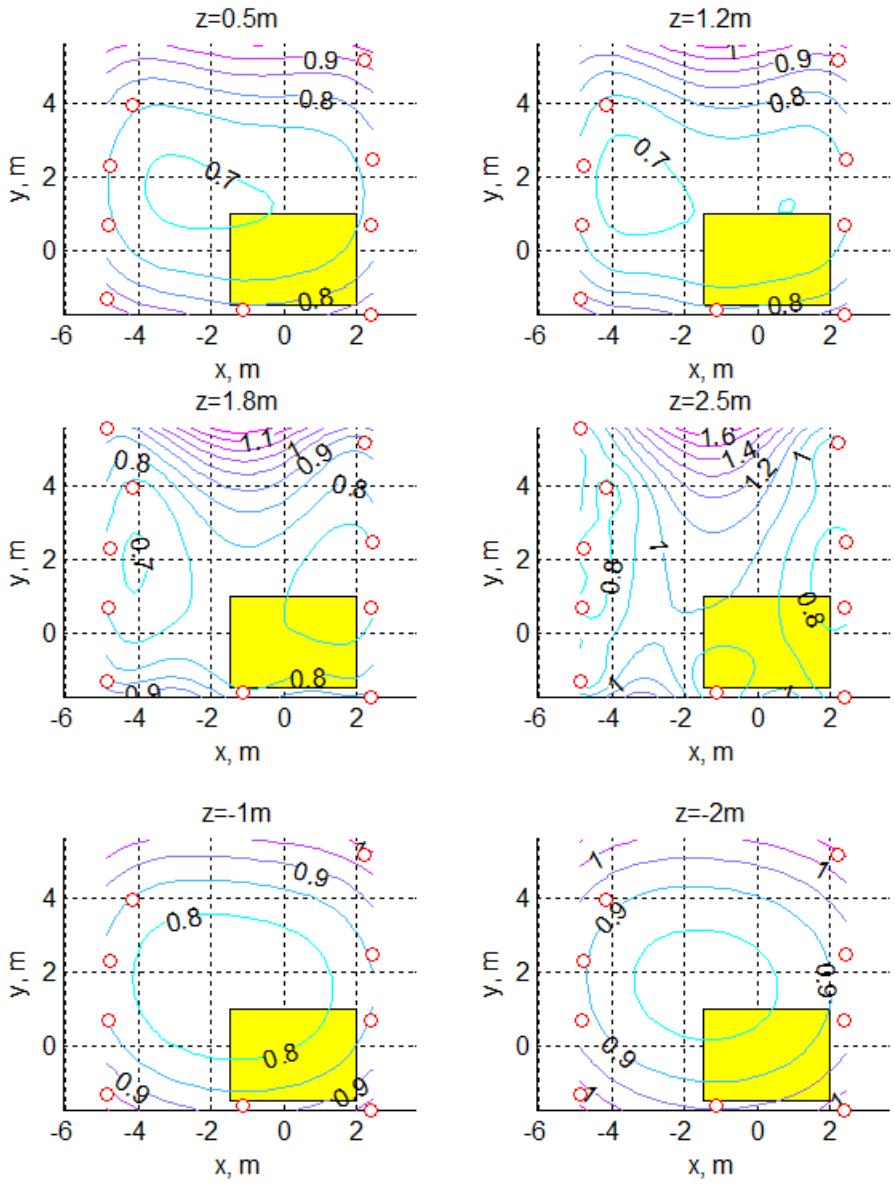


Figure 38: Isolines of DOP at Different Heights for the Case of Two Optitrack Cameras at Each ASEIL Camera Location.

It should be noted that thus far, we have considered an ideal rather than a practical case as we have not taken the FOV of the cameras into account. Figure 36 to Figure 38 were obtained without accounting for the cameras' FOV. Accounting for a 46° horizontal FOV of the V100:R2

Optitrack cameras (and 30° vertical FOV) leads to worsening of the DOP and shrinking further the flyable zone.

Before proceeding with the analysis of the limited FOV effect, consider one of the scenarios where two Qball-X4 quadrotors exchange places while avoiding some simulated obstacle as shown in Figure 39. For the case of unlimited FOV, the GDOP of both the Qball-X4 quadrotors while flying almost up to the ceiling, 2.8 m, is shown in Figure 40. We will use these results as our benchmark. As expected, most of the errors occur in the vertical channel. The bottom-most graph in Figure 40 shows the ratio between the vertical error to horizontal error. The vertical errors are twice to eight times larger than the horizontal error.

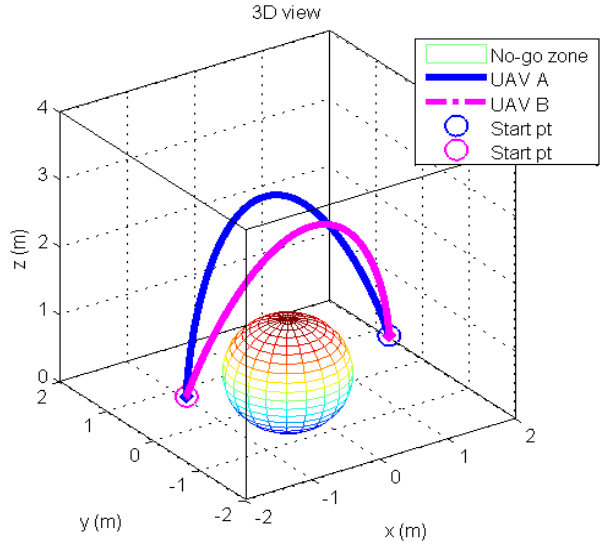


Figure 39: 3D Trajectory of Two Qballs Exchanging Places while Avoiding a Spherical Obstacle Placed at the Center.

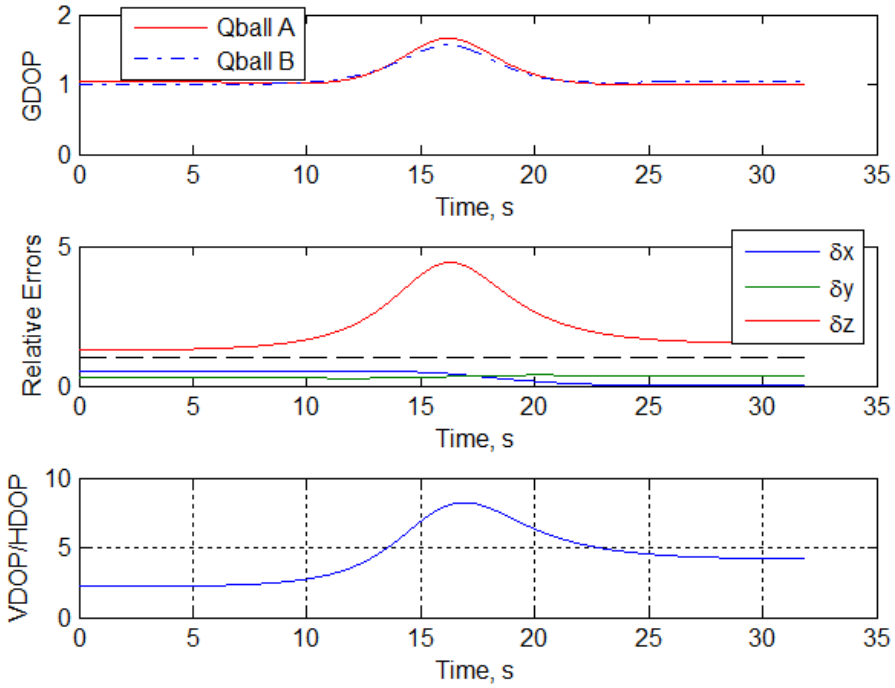


Figure 40: Change in DOP for a Qball-X4 Flying the 3D Trajectory (see Figure 39) in an Ideal (Unlimited FOV) 10-camera ASEIL Setup.

Figure 41 represents a more realistic DOP estimate, as compared to Figure 36, by taking into account the limited FOV of the cameras. As expected, the drastic change is caused by the number of visible cameras at each particular location in the lab as shown in Figure 42. It was observed that all ten cameras are centered at a point, which is approximately 1 m above the floor at the center of the coordinate frame established for the flyable zone.

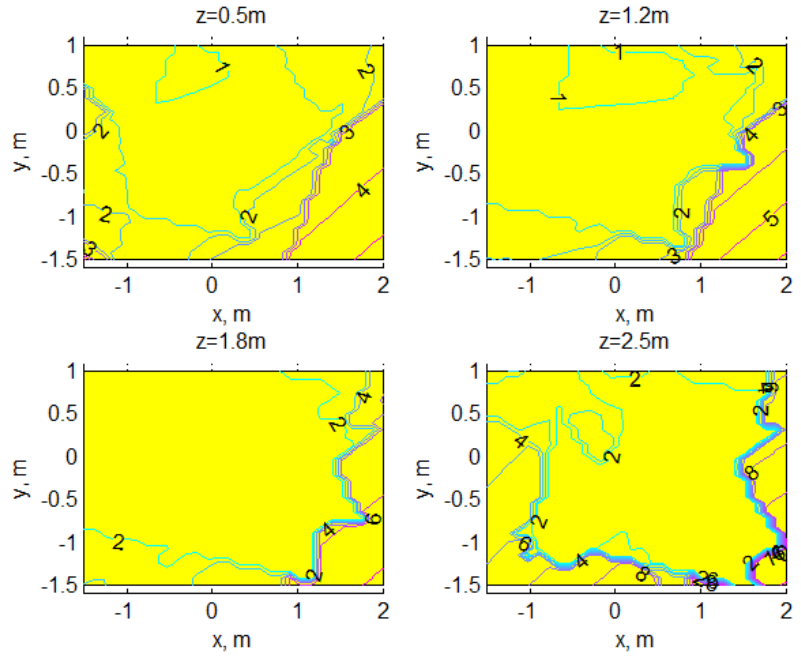


Figure 41: Isolines of DOP at Different Heights for a 10-camera ASEIL Setup Accounting for the Cameras' FOV.

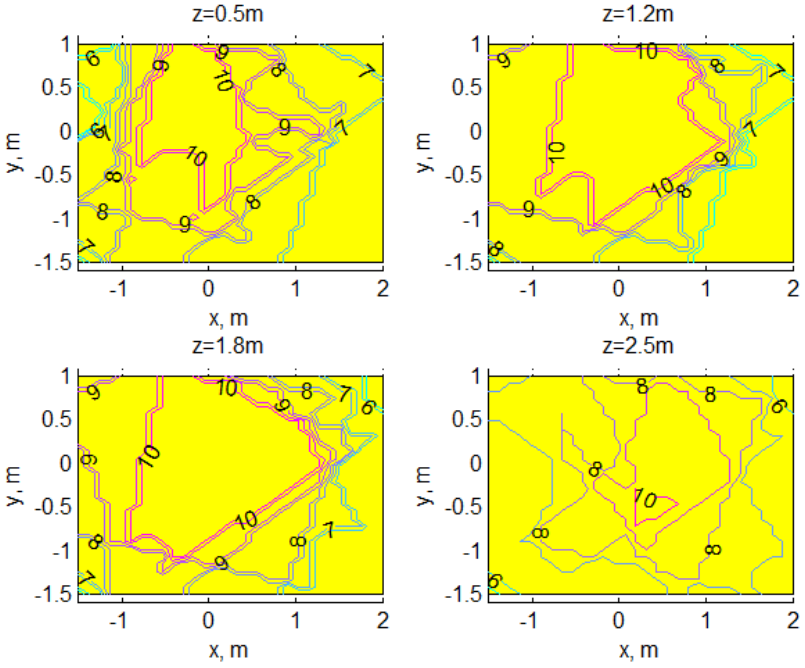


Figure 42: Isolines of Visible Cameras at Different Heights for a 10-camera ASEIL Setup.

Figure 43 features three plots, similar to those of Figure 40, for the more realistic case. For the realistic case, it can be observed that the DOP suffers a deeper degradation.

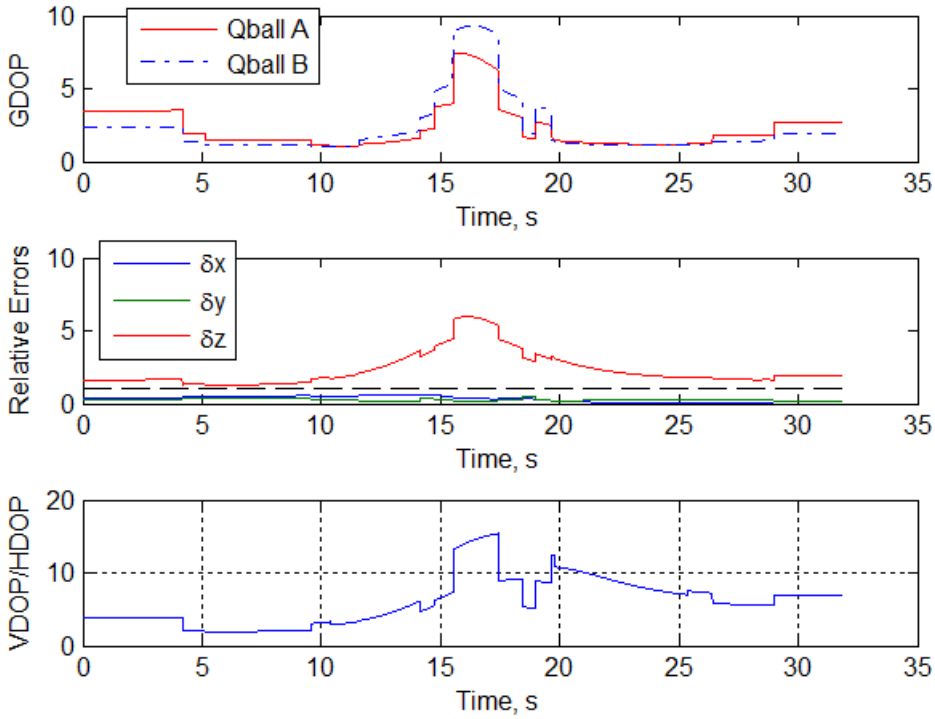


Figure 43: Change in DOP for a Qball-X4 Flying the 3D Trajectory (see Figure 39) in the Current 10-camera ASEIL Setup.

Doubling the number of cameras for the realistic case (with limited cameras' FOV) with the second set of cameras oriented exactly the same as the existing cameras, as shown in Figure 44, does not improve the situation as much as when the cameras' FOV is doubled. The cameras' FOV can be doubled by placing the second set of cameras at about the same location as the first set but orienting both sets of cameras to achieve a wider FOV as shown in Figure 45.

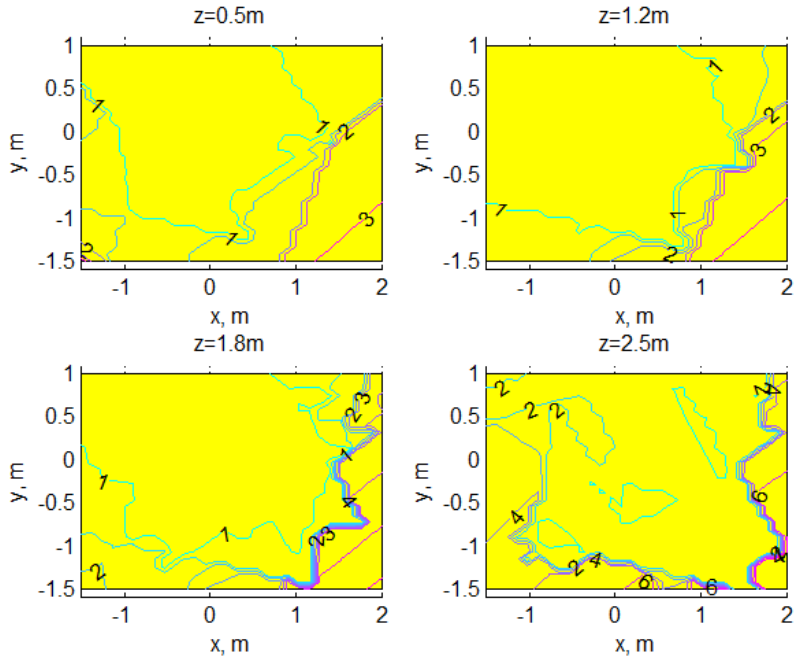


Figure 44: Isolines of DOP at Different Heights if the Number of Cameras is Doubled.

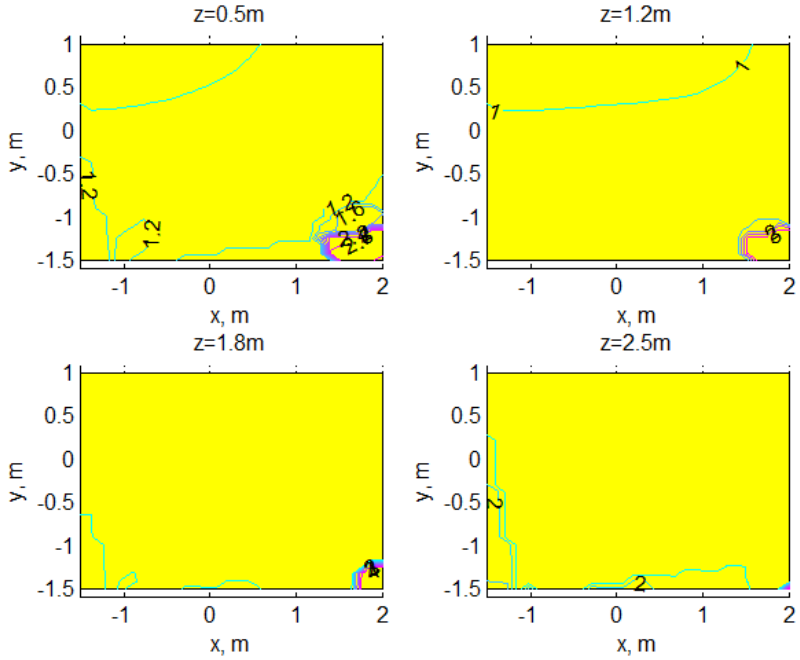


Figure 45: Isolines of DOP at Different Heights if the FOV Angle is Doubled.

Figure 46 shows that when the FOV are doubled, a better coverage of the entire flyable zone can be achieved.

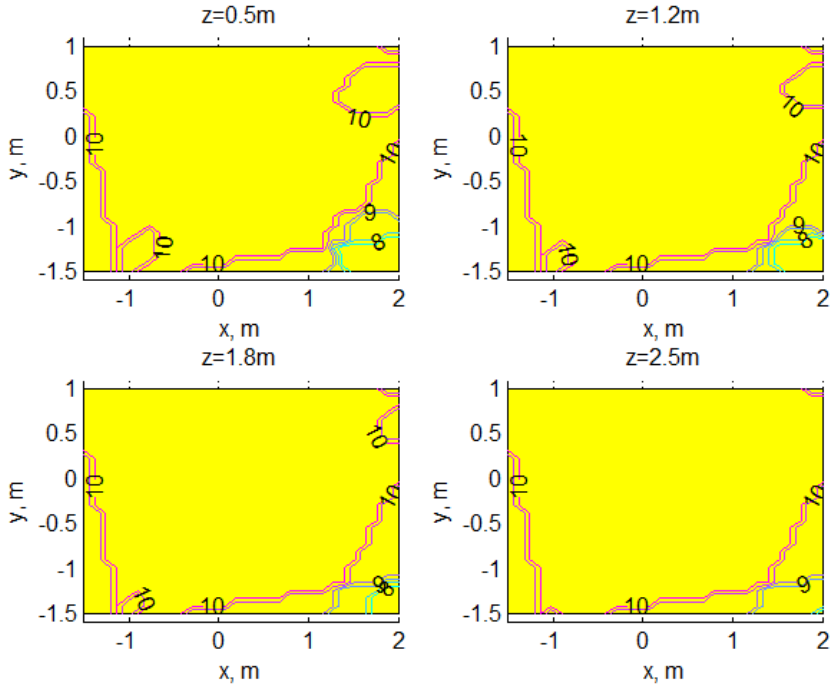


Figure 46: Number of Cameras Visible at Different Locations when the FOV Angle is Doubled.

Figure 47 shows the DOP for two Qball-X4 quadrotors flying on the same trajectories as those shown in Figure 39, which gives lower and smoother results.

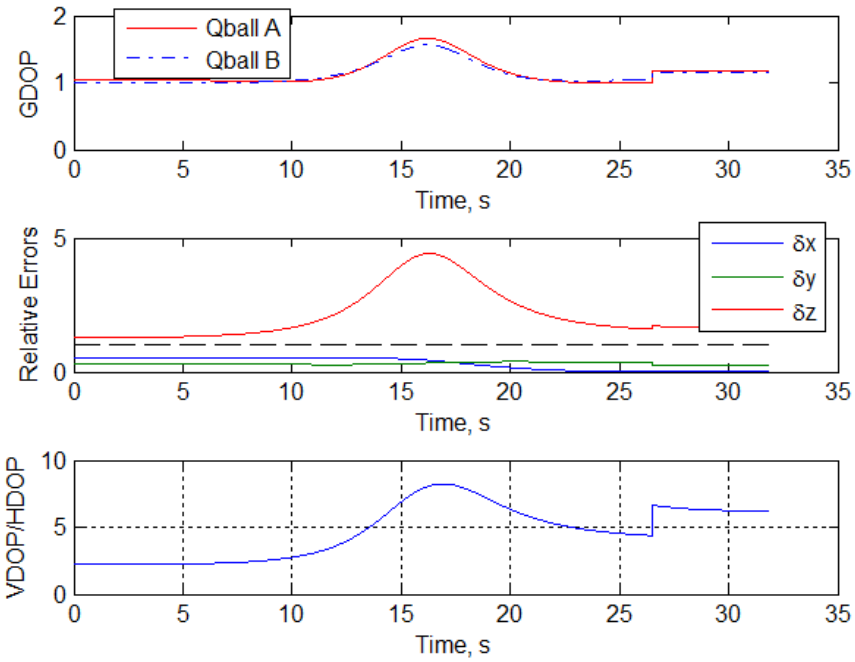


Figure 47: Change in DOP for a Qball-X4Quadrotor Flying a Typical 3D Trajectory in a Hypothetical 20-camera Doubled FOV ASEIL Setup.

It is understood that the geometrical configuration of the Optitrack camera constellation can affect the DOP; let us now briefly discuss the expected magnitudes of the tracking error. Figure 48 shows a setup of two tests that were conducted to estimate the Optitrack tracking error affected by a nonlinear distribution of the DOP within the workspace.

For the first test, a Qball-X4 quadrotor was placed on the floor close to the origin of the local tangent plane (LTP) and then manually moved along the z-coordinate as shown in Figure 48(a) by 1 cm for 20 intervals. Figure 49(b) shows the Optitrack tracking errors in two dimensions. The relative error between two increments stays about ±2 mm, while the overall error for the 20 increments increased to about 5 cm.

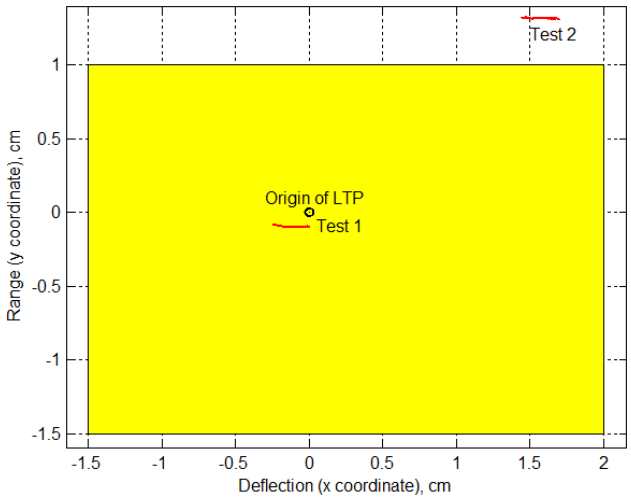


Figure 48: Plan View of the Workspace with the Locations of the Two Test Setups Marked.

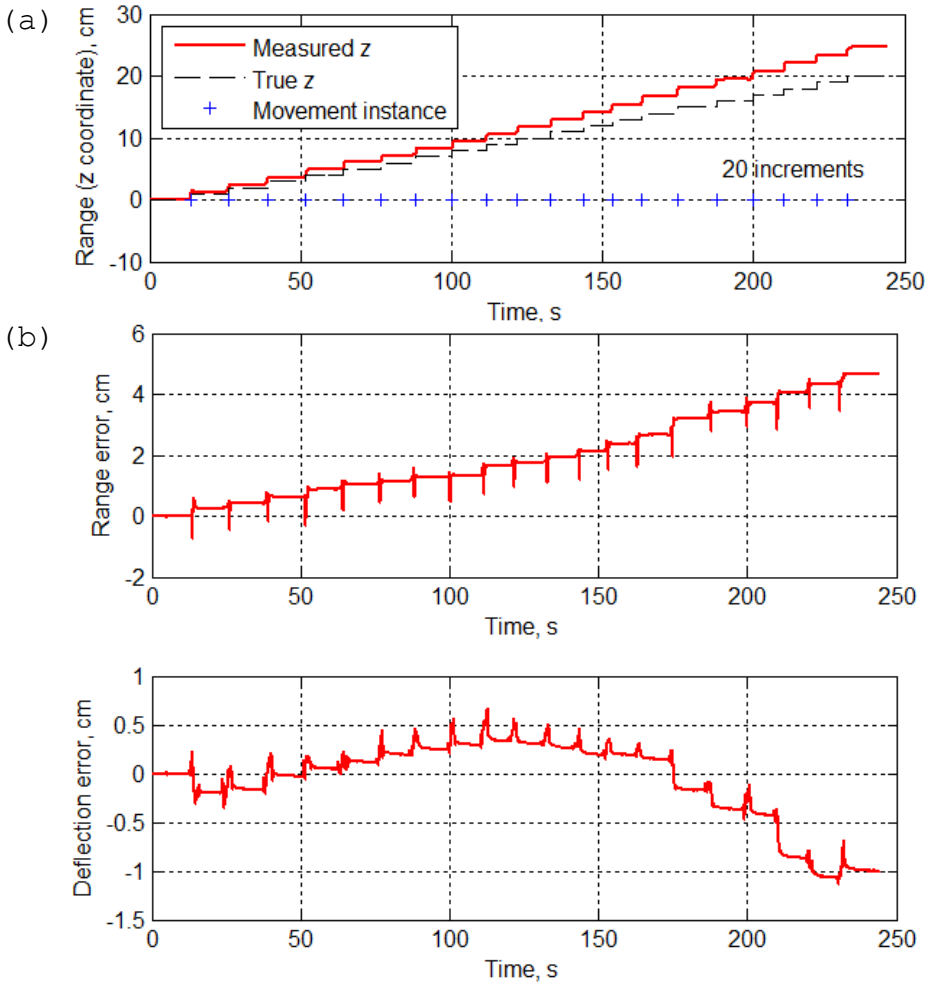


Figure 49: (a) Measured versus True Range (Test 1), (b) Optitrack Measurement Errors (Test 1).

A similar test (Test 2) was conducted at a different location as shown in Figure 48. With a poorer DOP at that location, the tracking error grew to approximately 6 cm. Comparing Figure 49(a) and Figure 50(a), it can be observed that the closer the Qball is to the origin of the LTP, the smaller the relative errors between the increments. The largest error between two consecutive increments (about 0.7 cm) is observed at the beginning of the second test.

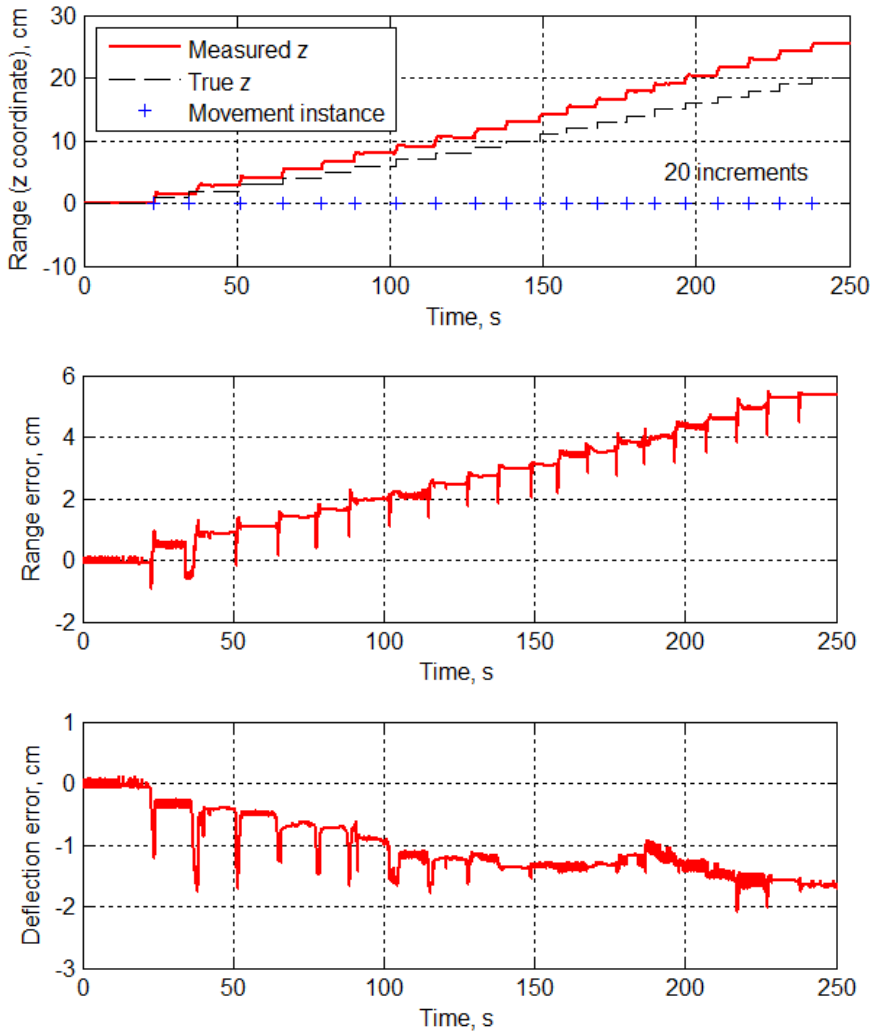


Figure 50: (a) Measured versus True range (Test 2), (b) Optitrack Measurement Errors (Test 2).

The altitude measurement errors are expected to be much larger. From the bottom-most plot in Figure 43, it can be noted that the error in the vertical channel can be 10 times larger than the horizontal error at floor level. These findings agree with the results obtained from actual tests; the error in the vertical channel as measured by the Optitrack system in midair is of the order of 0.4 m. Thus, an ultrasonic sensor, which assures about 1 cm error, is used for altitude control of the Qball-X4 instead of using the Optitrack system which gives larger errors in the vertical channel.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. SIMULATED AND ACTUAL FLIGHT DATA

A. OVERVIEW

In this chapter, the simulated results are compared in detail to the actual flight data. The test plan is first described. The position control and altitude control performances using the default controllers provided by Quanser are assessed. Also discussed here are the velocity limits, which were adjusted to observe whether there is any impact on the flight performance.

B. SENSORS RESOLUTION

The resolution for the sensors are as follows (Quanser 2011):

- 3-axis Accelerometer 3.33 mg/LSB
- 3-axis Gyroscopes 0.0125°/s/LSB
- 3-axis Magnetometer 0.5mGa/LSB
- Sonar 1 cm
- Optitrack 1 cm (best)

C. TEST PLAN DESCRIPTION

1. The Qball-X4 hovers at an altitude of 0.5 m and then executes an inverted L-shaped flight profile.

2. The Qball-X4 climbs to an altitude of 0.5 m, increases to 1.0 m and 1.5 m, then returns to 1.0 m and 0.5 m before landing.

3. The Qball-X4 performs a 120° heading (counter clockwise) turn, followed by a -120° heading (clockwise) turn.

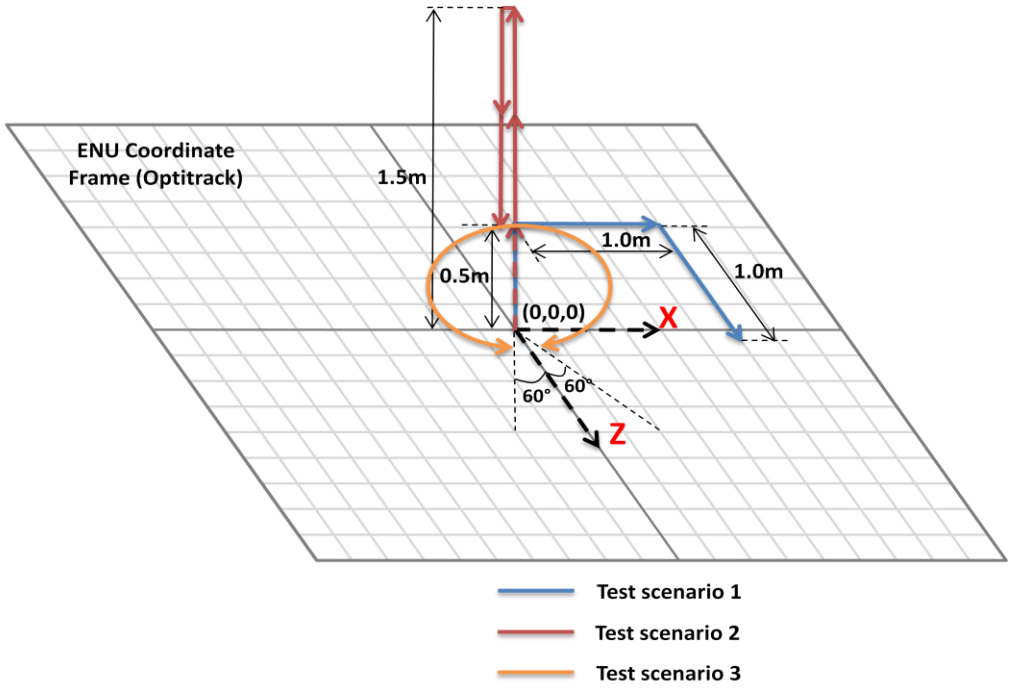


Figure 51: Illustration of Test Scenarios.

D. DEFAULT PARAMETER VALUES

Table 8 provides the list of the default saturator limits and gain values.

Table 8: List of Saturation Limits and Gains Values.

Parameter	Symbol	Values	Units
Saturation Limits:			
Roll/Pitch PWM Limit	Climit	0.025	% 20ms d.c.
Velocity Limit	vlimit	0.3	m/s
Height Velocity Limit	Vlimith	0.1	m/s
Roll/Pitch Limit	tlimit	0.0873	rad
Gains:			
Roll/Pitch Control Channel			
P-gain (outer loop)	Kp	0.7988	--
I-gain (outer loop)	Ki	0.1	--
D-gain (outer loop)	Kd	0.6901	--
Heading Control Channel			
P-gain (outer loop)	Kpyaw	0.0316	--
D-gain (outer loop)	Kdyaw	0.015	--
Height Control Channel			
P-gain (outer loop)	Kph	0.0062	--
I-gain (outer loop)	Kih	0.0032	--
D-gain (outer loop)	Kdh	0.006	--

E. TEST SCENARIO 1

1. Ground Track

Figure 52 shows the actual and simulated ground track. It can be seen that the quadrotor tracks reasonably well with the commanded trajectory, with a maximum cross-track error of approximately 0.1 m (10% of commanded value).

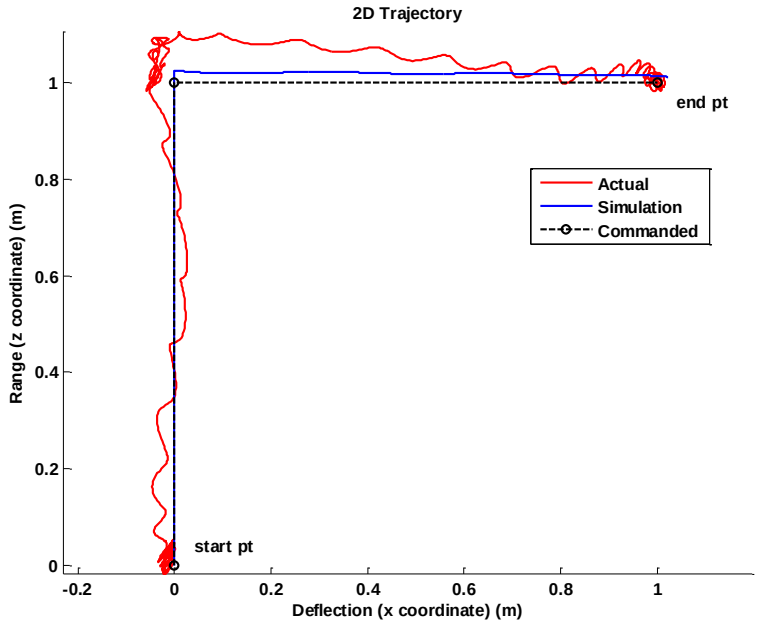


Figure 52: Plot of Actual and Simulated Ground Track.

2. X and Z Position

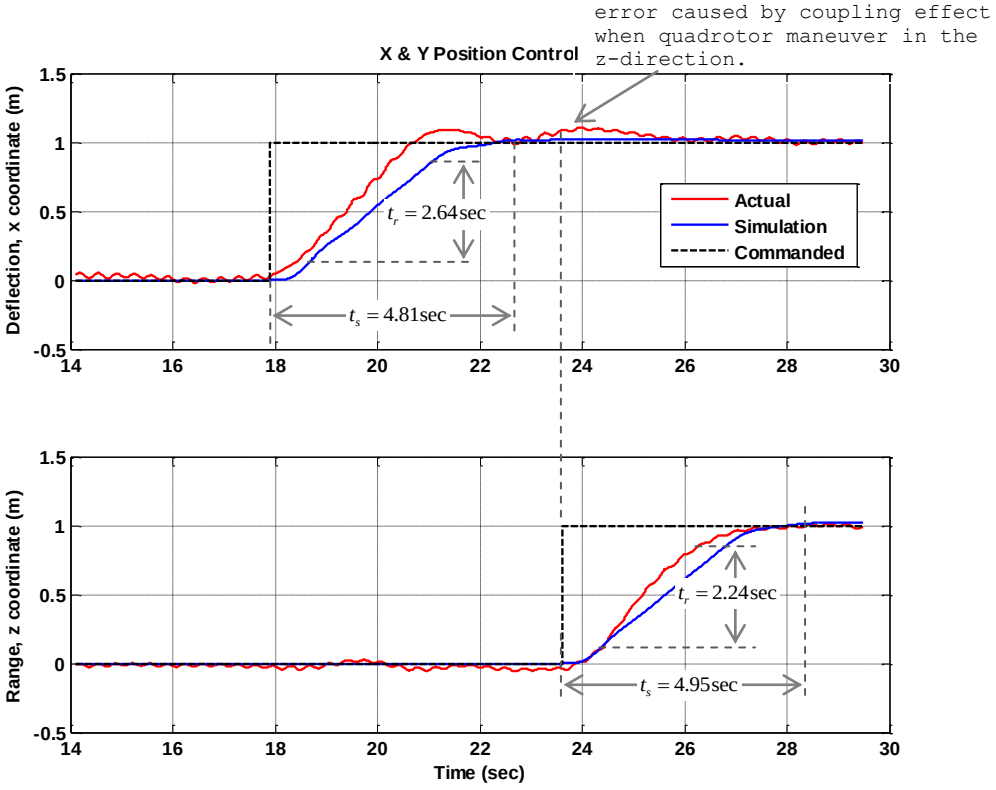


Figure 53: Plot of Actual and Simulated X and Z Position.

3. Height

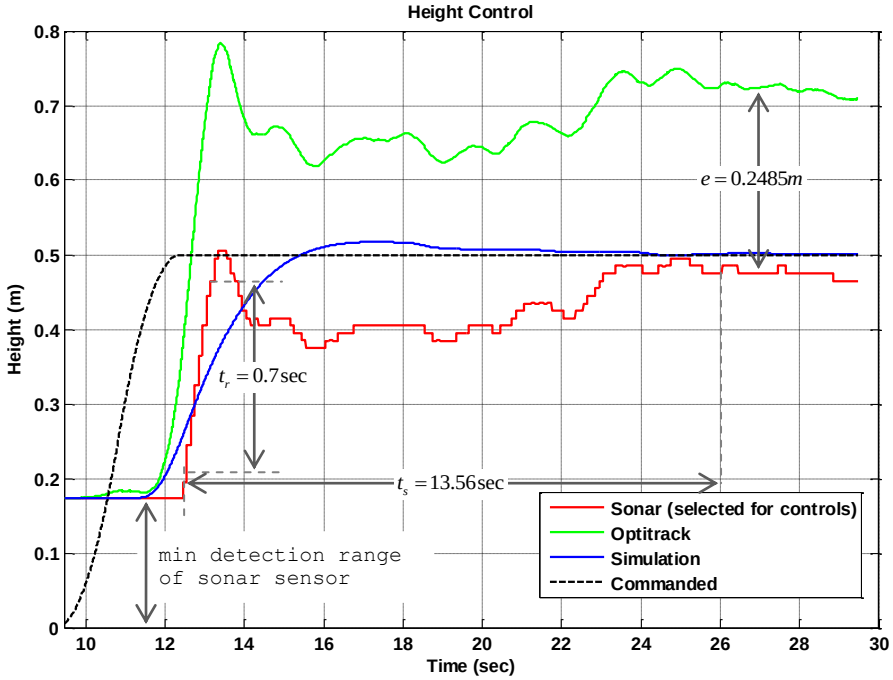


Figure 54: Plot of Actual and Simulated Heights.

4. Accelerations

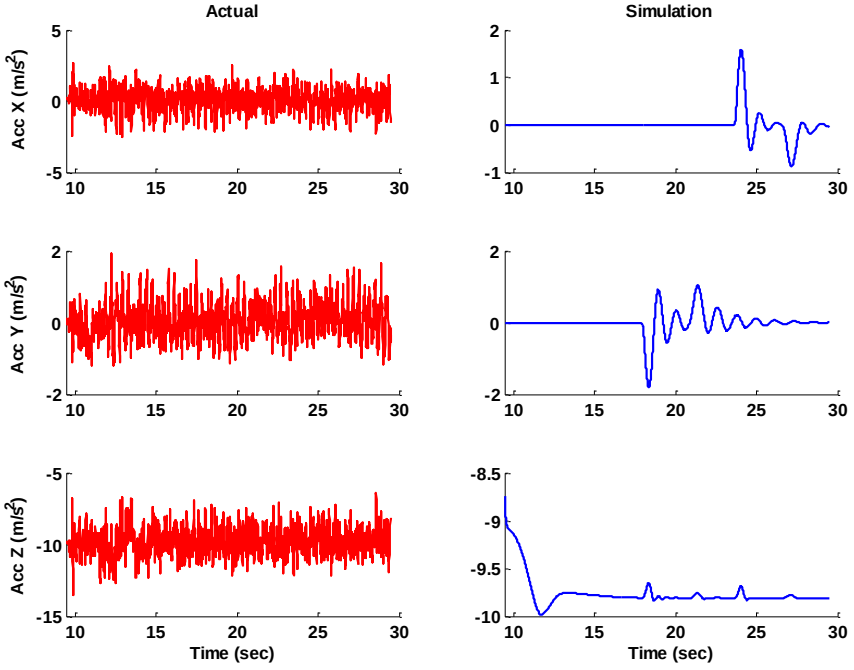


Figure 55: Plot of Actual and Simulated Accelerations.

5. Angular Rates

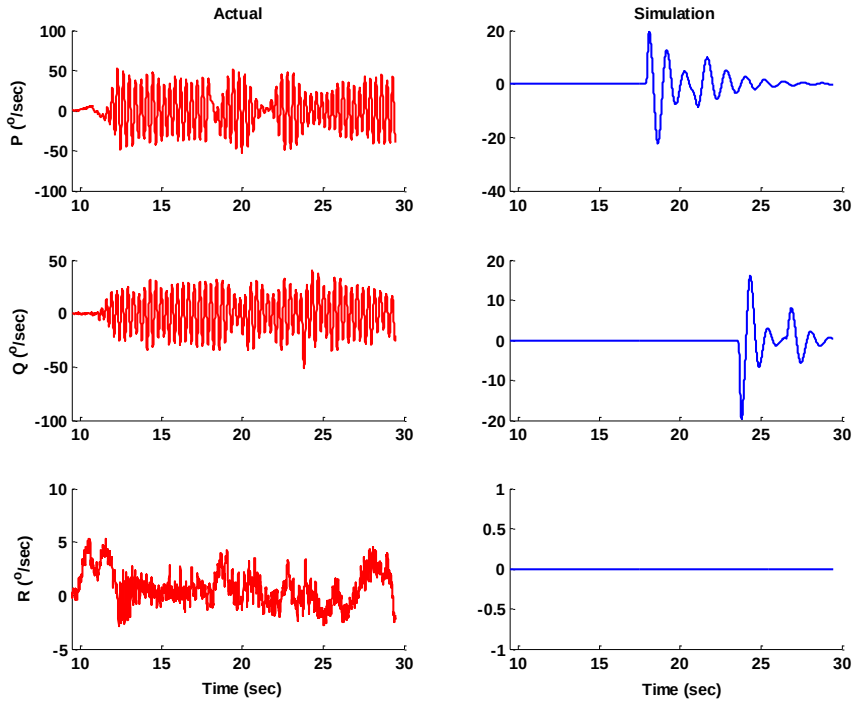


Figure 56: Plot of Actual and Simulated Angular Rates.

6. Euler Angles

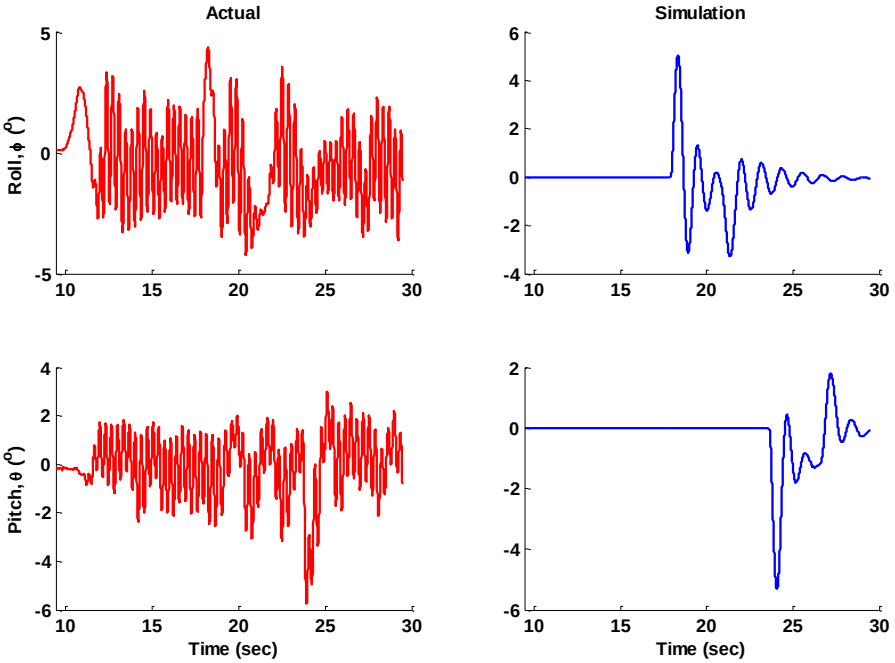


Figure 57: Plot of Actual and Simulated Euler Angles.

F. TEST SCENARIO 2 (HEIGHT INCREMENT)

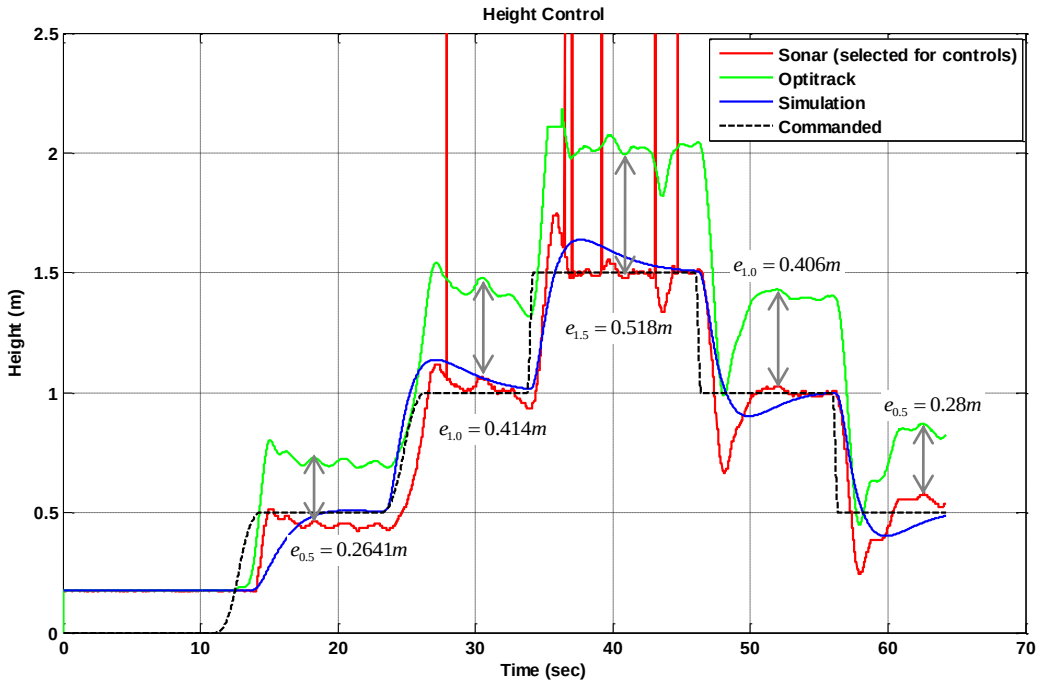


Figure 58: Plot of Actual and Simulated Incremental Heights.

It can be observed from Figure 58 that the noises associated with the sonar sensor get larger as the altitude of the quadrotor increases, while the altitude information from the Optitrack motion capture system is less noisy. However the altitude error from the Optitrack system increases with altitude. If we are able to create a comprehensive correction table for the Optitrack altitude data, it can be used in place of the sonar sensor for altitude control in the future for indoor experiments (to take advantage of its reduced noise). Also, notice that the overshoot during descent tends to be larger than that during ascent.

To make the altitude information obtained from the Optitrack system useful, the following methods were used.

- At discrete altitude intervals $\left(h_{cmd} = h < 0.5,\ 0 \le h < 0.5,\ 0.5 \le h < 1.0, \ldots\right)$ and when the sonar and $y-$Optitrack reached steady-state, collect all the steady-state data points and perform averaging. The greater the number of discrete intervals and data points, the better the accuracy.
- Obtain a mapping factor for each discrete interval, where the mapping factor is given by:

$$K_{map}\Big|_{h<0.5,\ 0\le h<0.5,\ 0.5\le h<1.0,\ \ldots} = \left[\frac{sonar_{ave}}{y\_optitrack_{ave}}\right]_{h<0.5,\ 0\le h<0.5,\ 0.5\le h<1.0,\ \ldots} \quad (46)$$

  - Produce a table to store all the mapping factors at each discrete interval.
  - Multiply the $y-$Optitrack data with the corresponding mapping factor, according to the interval into which it falls.

$$y\_optitrack_{(i)} = K_{map(i)} sonar_{(i)} \quad (47)$$

Figure 59 shows the result of the corrected $y-$Optitrack information after multiplying with the appropriate mapping factor.

Mapping Equation:

$$K_{map}\Big|_{h<0.5,\ 0\le h<0.5,\ 0.5\le h<1.0,\ \ldots} = \left[\frac{sonar_{ave}}{y\_optitrack_{ave}}\right]_{h<0.5,\ 0\le h<0.5,\ 0.5\le h<1.0,\ \ldots}$$

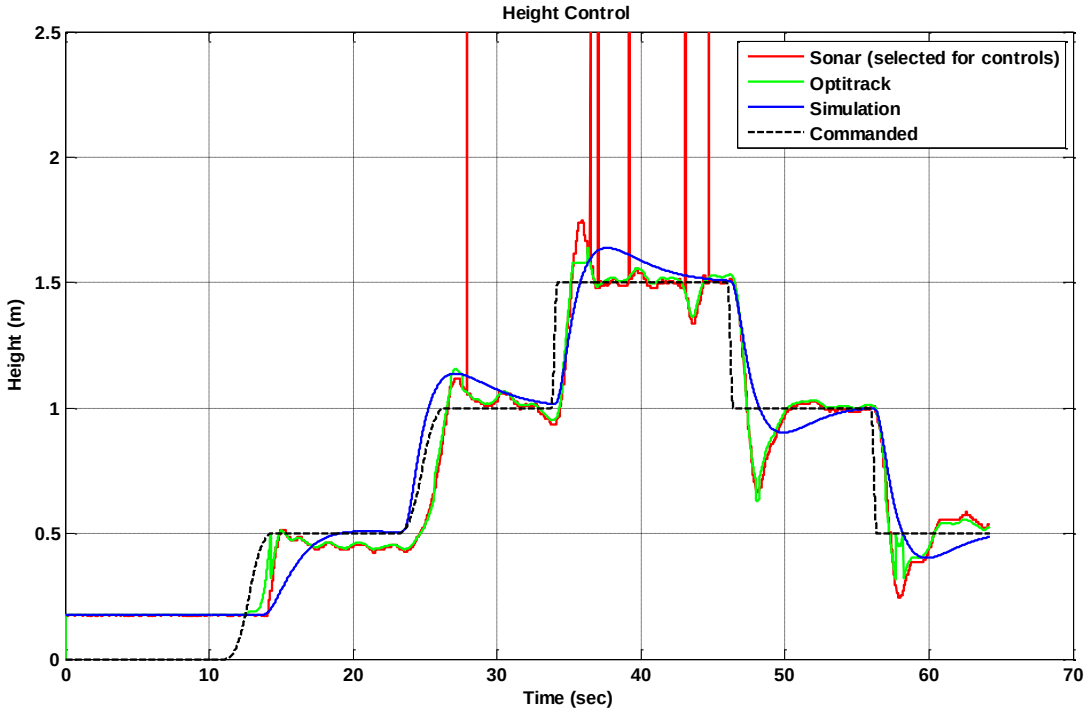


Figure 59: Optitrack Altitude After Corrected with Appropriate Mapping Function.

G. TEST SCENARIO 3 (HEADING CONTROL)

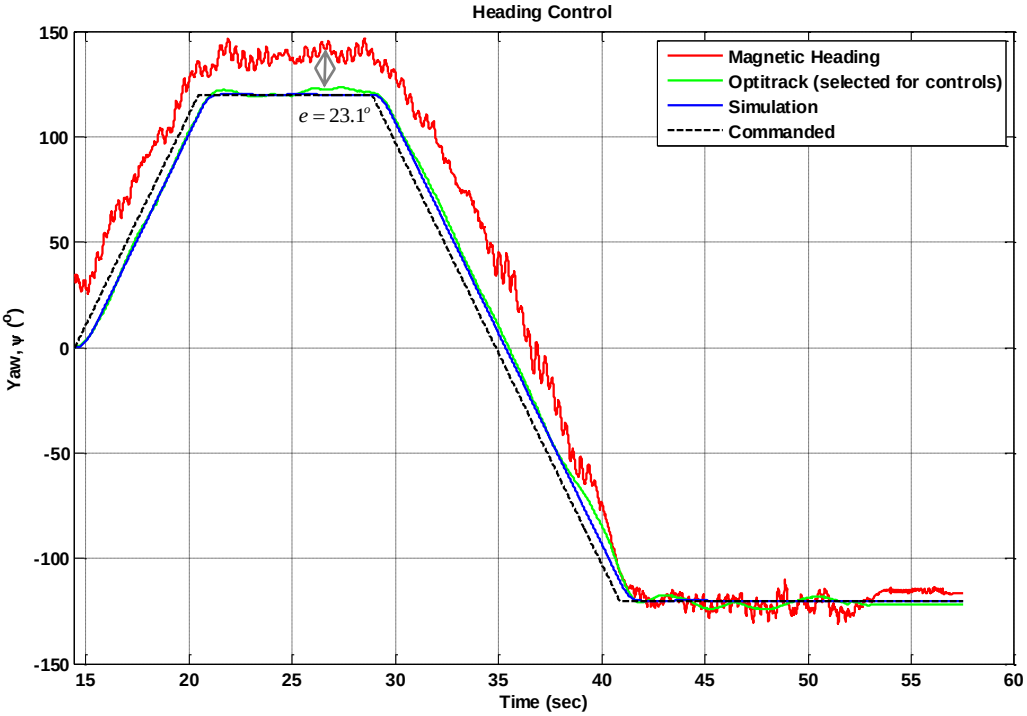


Figure 60: Plot of Actual and Simulated Heading.

From Figure 60, it can be seen that the Optitrack heading tracks the commanded value very closely, while there are errors associated with the magnetic heading when the heading are positive but gradually reduces as the heading turns negative. This indicates a need to recalibrate the magnetometer onboard the Qball-X4 quadrotor if we are to use the sensor information for the heading feedback control.

H. VELOCITY LIMITS TEST

The velocity saturation limits were adjusted, and the impact on the ground track performance was observed. The

limits were varied between 0.2 to 0.9, with the default value being 0.3.

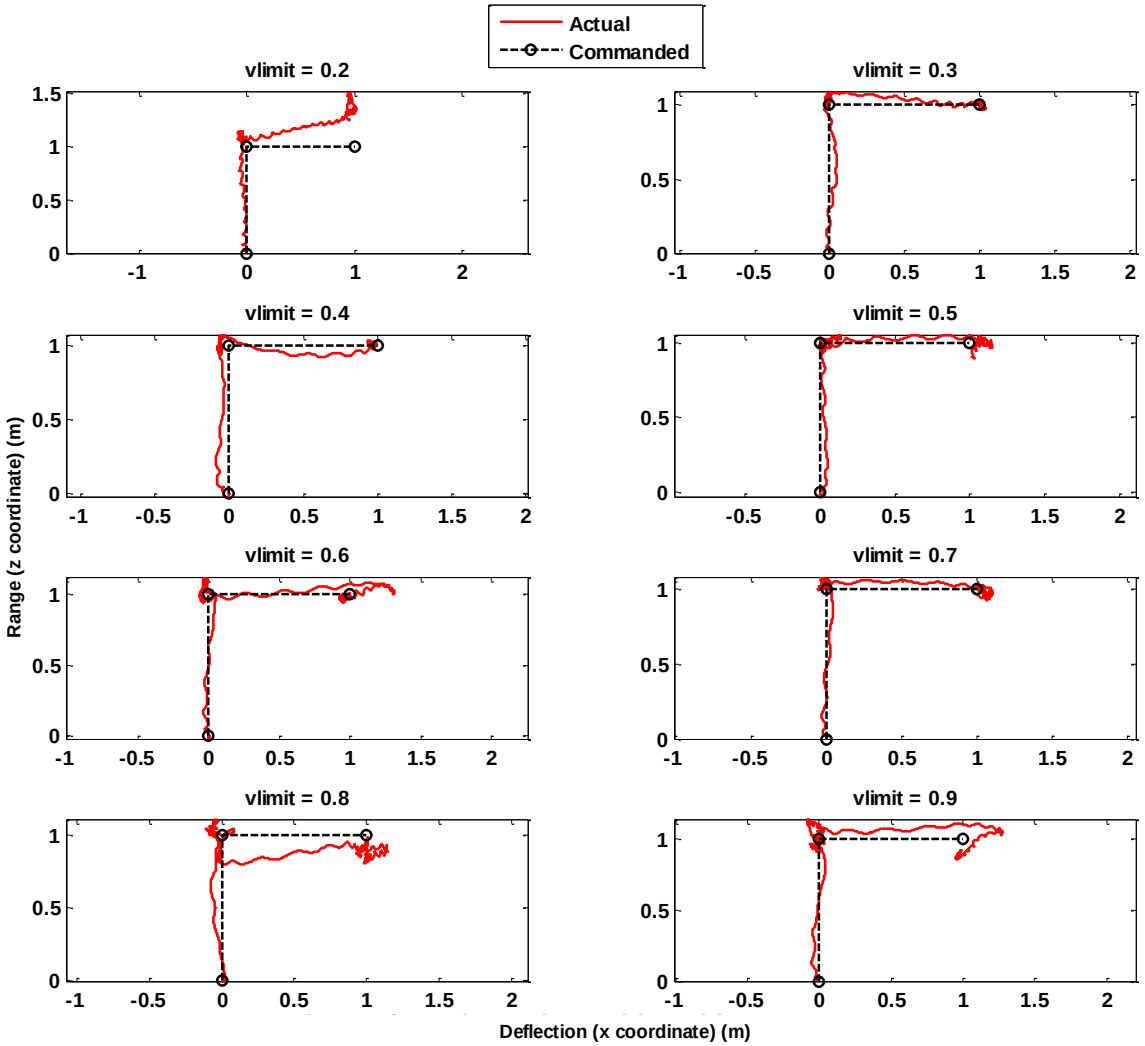


Figure 61: Plots of Ground Track with Variation in vlimits.

From Figure 61, it can be observed that there was no strong indication that position tracking improves with higher velocity limits. In fact, reduction in the velocity limit to 0.2 resulted in poorer performance, likely due to lower control effectiveness caused by lower gain values.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. DIRECT METHOD USING INVERSE DYNAMICS IN VIRTUAL DOMAIN

A. INTRODUCTION

In Chapter 0, a mission scenario was created for the Qball-X4 quadrotor. The quadrotor is to perform an obstacle collision avoidance maneuver while using the direct method of calculus of variations exploiting the inverse dynamics in virtual domain (IDVD) in solving for the trajectory optimization problem.

The IDVD method is the preferred choice for several reasons. Firstly, this method allows for the satisfaction of higher-order derivatives at both the initial and final points (allowing for very smooth transition to a newly generated trajectory). Second, it permits the use of any model and performance index, such that it is not subjected to the curse of dimensionality and does not require differentiability of the performance index. Finally, it requires significantly less optimizing parameters (i.e., typically fewer than 10) compared to other direct methods; thus, it greatly reduces the computational time required to generate a feasible trajectory, allowing for real-time trajectory generation onboard the quadrotor during flight (Yakimenko 2010).

The 6DOF model of the quadrotor is already described in Chapter III. Following from there, the general architecture of the autonomous control system is first introduced. A conventional PID controller is used for trajectory following. Then the trajectory optimization problem is formulated along with the detailed numerical

trajectory optimization routine. The outcome of the simulation is presented in Chapter VII.

B. CONTROLLER ARCHITECTURE

The proposed control system architecture (O. Yakimenko 2010) is presented in Figure 62. The top section represents the common feedback control for path tracking by the quadrotor while the bottom section provides periodic updates of the trajectory by the trajectory generator. The trajectory generator computes a quasi-optimal route in a relatively short time (typically on the order of 10 to 100s) making it possible for re-optimization of the trajectory during flight. This is crucial in the event of unexpected obstruction along the original planned path. The interpolator produces samples of the reference trajectory at the desired high frequency rate required by the controller.

During the mission, there might be a need to modify the mission scenario. When the discrepancy between the current and desired state becomes too large (i.e., due to wind or noise disturbances), for instance, the update switch triggers the trajectory generator to re-compute a new quasi-optimal trajectory, taking the current state as the new vector of initial conditions.

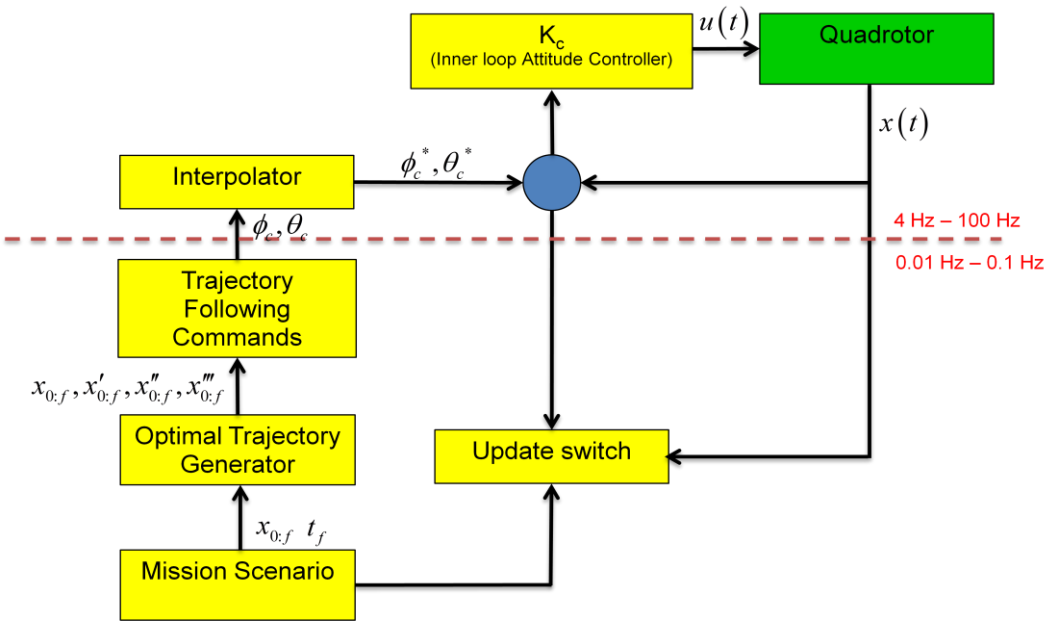


Figure 62: Proposed Controller Architecture (After O. Yakimenko 2010).

C. TRAJECTORY OPTIMIZATION

This section aims to find the reference trajectory x_{ref} and desired control profile u_{ref} by solving the trajectory optimization problem, utilizing the differential flatness characteristics of the quadrotor dynamics. This is followed by a detailed optimization routine.

1. Differential Flatness and Optimal Problem Formulation in Output Space

The differential flatness property of a system refers to the possibility of expressing its states and control vectors in terms of the output vectors and their derivatives (Chelouah 1997).

From Eqn.(17), the components of the control vector u can be easily expressed in terms of the states and their derivatives:

$$u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} F_{T1} + F_{T2} + F_{T3} + F_{T4} \\ (F_{T3} - F_{T4})l \\ (F_{T1} - F_{T2})l \\ (\tau_3 + \tau_4 - \tau_1 - \tau_2) \end{bmatrix} = \begin{bmatrix} m\sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} - g)^2} \\ J_{xx}\ddot{\phi} \\ J_{yy}\ddot{\theta} \\ J_{zz}\ddot{\psi} \end{bmatrix} \tag{48}$$

Expressing Eqn.(12) in the following form shown in Eqn.(49)

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \frac{1}{m} \begin{bmatrix} u_1 \sin\phi \sin\psi + u_1 \cos\phi \sin\theta \cos\psi \\ -u_1 \sin\phi \cos\psi + u_1 \cos\phi \sin\theta \sin\psi \\ u_1 \cos\phi \, cos\theta - mg \end{bmatrix} \tag{49}$$

To simplify Eqn.(49), we assume the heading angle ψ to be very small, such that the rotational part of the state vector can then be expressed (in terms of the output vector and its derivatives) as:

$$\begin{aligned} \theta &= \tan^{-1}\left(\frac{\ddot{x}}{\ddot{z} + g}\right) \\ \phi &= \sin^{-1}\left(-\frac{\ddot{y}}{\sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} - g)^2}}\right) \end{aligned} \tag{50}$$

Singularities in Eqn.(50) can occur when $\ddot{z} = -g$; that is, when the quadrotor is experiencing a free-fall. To avoid this, we add the constraints $u_1 > 0$, $\phi < 90^o$ and $\theta < 90^o$.

Taking the differentiation of Eqn.(50) using the quotient rule and trigonometric function, we can perform the following.

Reviewing the differentiation of the trigonometric function:

$$\frac{d}{dt}\left(\tan^{-1} f(t)\right) = \frac{1}{f^2(t)+1} f'(t)$$
$$\frac{d}{dt}\left(\sin^{-1} f(t)\right) = \frac{1}{\sqrt{1-f^2(t)}} f'(t)$$

Reviewing the differentiation using the quotient rule:

$$\frac{d}{dt}\left\{\frac{f(t)}{g(t)}\right\} = \frac{g(t)f'(t) - f(t)g'(t)}{\{g(t)\}^2}$$

Derive $\dot{\theta}$, such that

$$\begin{aligned}\dot{\theta} &= \frac{1}{\left(\frac{\ddot{x}}{\ddot{z}+g}\right)^2+1}\frac{d}{dt}\left(\frac{\ddot{x}}{\ddot{z}+g}\right) \\ &= \frac{1}{\left(\frac{\ddot{x}}{\ddot{z}+g}\right)^2+1}\left(\frac{(\ddot{z}+g)\frac{d}{dt}(\ddot{x}) - \ddot{x}\frac{d}{dt}(\ddot{z}+g)}{(\ddot{z}+g)^2}\right) \\ &= \frac{\dddot{x}(\ddot{z}+g) - \ddot{x}\dddot{z}}{\ddot{x}^2+(\ddot{z}+g)^2}\end{aligned} \qquad (51)$$

and $\dot{\phi}$ can also be derived as:

$$\dot{\phi} = \frac{1}{\sqrt{1-\left(-\frac{\ddot{y}}{\sqrt{\ddot{x}^2+\ddot{y}^2+(\ddot{z}-g)^2}}\right)^2}}\frac{d}{dt}\left\{-\frac{\ddot{y}}{\sqrt{\ddot{x}^2+\ddot{y}^2+(\ddot{z}-g)^2}}\right\}$$

$$= \frac{1}{\sqrt{1-\left(-\frac{\ddot{y}}{\sqrt{\ddot{x}^2+\ddot{y}^2+(\ddot{z}-g)^2}}\right)^2}}\left\{\frac{\left(\sqrt{\ddot{x}^2+\ddot{y}^2+(\ddot{z}-g)^2}\right)\frac{d}{dt}(-\ddot{y}) - (-\ddot{y})\frac{d}{dt}\left(\sqrt{\ddot{x}^2+\ddot{y}^2+(\ddot{z}-g)^2}\right)}{\left(\sqrt{\ddot{x}^2+\ddot{y}^2+(\ddot{z}-g)^2}\right)^2}\right\}$$

$$= \frac{\dddot{y}\left\{\dddot{x}\ddot{x} + \dddot{y}\ddot{y} + \dddot{z}(\ddot{z}-g)\right\} - \ddot{y}\left\{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z}-g)^2\right\}}{\left\{\ddot{x}^2 + \ddot{y}^2 + \left(\ddot{z}-g\right)^2\right\}\left\{\sqrt{\ddot{x}^2 + \left(\ddot{z}-g\right)^2}\right\}} \tag{52}$$

Eqn.(51) and Eqn.(52) can be differentiated once more, and the results substituted into Eqn.(48),

$$\begin{aligned}\ddot{\theta} &= \frac{\left\{\ddot{x}^2 + \left(\ddot{z}+g\right)^2\right\}\frac{d}{dt}\left\{\dddot{x}\left(\ddot{z}+g\right) - \dddot{x}\ddot{z}\right\} - \left\{\dddot{x}\left(\ddot{z}+g\right) - \dddot{x}\ddot{z}\right\}\frac{d}{dt}\left\{\ddot{x}^2 + \left(\ddot{z}+g\right)^2\right\}}{\left\{\ddot{x}^2 + \left(\ddot{z}+g\right)^2\right\}^2} \\ &= \frac{\ddddot{x}\left(\ddot{z}+g\right) - \dddot{x}\dddot{z}}{\ddot{x}^2 + \left(\ddot{z}+g\right)^2} - \frac{2\left\{\dddot{x}\ddot{x} + \dddot{z}\left(\ddot{z}+g\right)\right\}\left\{\dddot{x}\left(\ddot{z}+g\right) - \dddot{x}\ddot{z}\right\}}{\left\{\ddot{x}^2 + \left(\ddot{z}+g\right)^2\right\}^2}\end{aligned} \tag{53}$$

and

$$\begin{aligned}\ddot{\phi} = &\frac{\left\{\ddot{x}^2 + \ddot{y}^2 + \left(\ddot{z}-g\right)^2\right\}\left\{\sqrt{\ddot{x}^2 + \left(\ddot{z}-g\right)^2}\right\}C_1}{\left\{\left\{\ddot{x}^2 + \ddot{y}^2 + \left(\ddot{z}-g\right)^2\right\}\left\{\sqrt{\ddot{x}^2 + \left(\ddot{z}-g\right)^2}\right\}\right\}^2} - \\ &\frac{\left\{\dddot{y}\left\{\dddot{x}\ddot{x} + \dddot{y}\ddot{y} + \dddot{z}(\ddot{z}-g)\right\} - \ddot{y}\left\{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z}-g)^2\right\}\right\}C_2}{\left\{\left\{\ddot{x}^2 + \ddot{y}^2 + \left(\ddot{z}-g\right)^2\right\}\left\{\sqrt{\ddot{x}^2 + \left(\ddot{z}-g\right)^2}\right\}\right\}^2}\end{aligned} \tag{54}$$

where

$$\begin{aligned}C_1 = &\dddot{y}\left\{\ddot{x}^2 + \ddot{y}^2 + \ddot{z}^2 + \dddot{x}\ddot{x} + \dddot{y}\ddot{y} + \left(\ddot{z}-g\right)\dddot{z}\right\} - \\ &\ddot{y}\left\{\dddot{x}\ddot{x} + \dddot{y}\ddot{y} + \left(\ddot{z}-g\right)\dddot{z}\right\} - \\ &\dddot{y}\left\{\ddot{x}^2 + \ddot{y}^2 + \left(\ddot{z}-g\right)^2\right\} \\ C_2 = &\frac{\left\{\ddot{x}^2 + \ddot{y}^2 + \left(\ddot{z}-g\right)^2\right\}\left\{\dddot{x}\ddot{x} + \left(\ddot{z}-g\right)\dddot{z}\right\}}{\sqrt{\ddot{x}^2 + \left(\ddot{z}-g\right)^2}} + \\ &2\sqrt{\ddot{x}^2 + \left(\ddot{z}-g\right)^2}\left\{\dddot{x}\ddot{x} + \dddot{y}\ddot{y} + \left(\ddot{z}-g\right)\dddot{z}\right\}\end{aligned}$$

Eqn.(53) and Eqn.(54) can be substituted into Eqn.(48).

The state vector x and control vector u can be expressed as some nonlinear function h_1 and h_2 as a function of the output vector y and its derivatives

$$\begin{aligned} x &= h_1(y, \dot{y}, ...) \\ u &= h_2(y, \dot{y}, ...) \end{aligned} \tag{55}$$

Expressing the optimization problem within the output space by taking advantage of the differentially flat characteristics of the quadrotor dynamics can significantly reduce the computation time for constraint handling since most constraints arise, for instance, from obstacle avoidance occurring in the output space.

Let the performance index for the obstacle collision avoidance be expressed as the following form:

$$J = (1-w)\frac{1}{t_f}\left(\int_0^{t_f}\sqrt{P_h\left(\dot{x}^2+\dot{y}^2\right)+P_v\dot{z}^2}\,dt + P_r\int_0^{t_f}\frac{V_r}{D^2}\,dt\right) + w\left(t_f - T\right)^2 \tag{56}$$

$$\begin{aligned} &\min_{y(t)} J(y(t)) \quad \text{for } t \in \left[0, t_f\right] \\ &s.t. \quad \dot{y}_0 - g_1(y(t_0)) = 0 \\ &\qquad \dot{y}_f - g_1(y(t_f)) = 0 \\ &\qquad c_1(y(t)) \le 0 \end{aligned} \tag{57}$$

where w, P_h and P_v are the weighting factors, t_f is the time of flight, V_r is the radial velocity, D is the distance from the quadrotor to the center of the obstacle, and T is the desired time-of-arrival.

Using a suitable parameterization of the output vector components, with some reference functions dependent on a few varied parameters, the boundary problem can be solved a

priori. As such, this eliminates the necessity to integrate differential equations, and the optimal problem can simply be formulated as:

$$\begin{aligned} &\min_{\xi} J(y(\xi)) \\ &s.t. \quad \mathrm{c}_1(y(\xi)) \le 0 \end{aligned} \tag{58}$$

where ξ is the vector of varied parameters

The optimal problem can be solved using the *fminsearch* or *fmincon* function in MATLAB.

2. Decoupling Space and Time

To decouple space and time, so as to allow independent optimization of the trajectory and speed profile, an abstract argument τ, also known as the virtual arc, is to be introduced. The argument τ, which is in the virtual domain, can be related back to time using the variable speed factor:

$$\lambda(\tau) = \frac{d\tau}{dt} \tag{59}$$

It should be noted that scaling the virtual speed profile $\lambda(\tau)$ does not really matter since higher values of λ will only result in larger τ_f, leaving other parameters in the time domain unchanged. Changing τ_f changes the shape of the candidate trajectory but does not affect the boundary conditions.

3. Parameterization

To reduce the optimal problem into a finite amount, a suitable parameterization is to be performed. The detailed procedure is explained as follows:

First, we assume that all these Cartesian coordinates follow some reference polynomial functions, where the order of the polynomials depends on the number of boundary conditions to be satisfied. The minimum degree of the polynomial is defined according to:

$$n = d_0 + d_f + 1 \tag{60}$$

where d_0, d_f are the maximum orders of the time derivative of the quadrotor coordinates at the initial and terminal points, respectively. It should be noted that other parameterization, such as that presented by Slegers and Yakimenko (Slegers and Yakimenko 2011), may also be used.

Thus, let the Cartesian coordinates (x, y, z) of the reference trajectory be represented by the following:

$$\begin{aligned} x(\tau) &= \sum_{k=0}^{n} a_k \frac{\{\max(1, k-2)\}!\tau^k}{k!} \\ x'(\tau) &= \sum_{k=1}^{n} a_k \frac{\{\max(1, k-2)\}!\tau^{k-1}}{(k-1)!} \\ x''(\tau) &= \sum_{k=2}^{n} a_k \tau^{k-2} \\ x'''(\tau) &= \sum_{k=2}^{n} (k-2) a_k \tau^{k-3} \end{aligned} \tag{61}$$

In the same manner, we define for y and z.

It is desirable for the trajectory at the terminal stage to be smoother; thus, we exploit the case where $d_f = 3$

with $x_0'' = 0,\ x_f'' = 0$ and the only varied parameter are thus τ_f, x_0''', x_f'''. Therefore, we are interested in the case where $d_0 = 3,\ d_f = 3$ and $n = d_0 + d_f + 1 = 7$.

The unknown coefficients in Eqn.(61) can then be found by solving the following matrix of algebraic equations:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & \tau_f & \frac{1}{2}\tau_f^2 & \frac{1}{6}\tau_f^3 & \frac{1}{24}\tau_f^4 & \frac{1}{60}\tau_f^5 & \frac{1}{120}\tau_f^6 & \frac{1}{210}\tau_f^7 \\ 0 & 1 & \tau_f & \frac{1}{2}\tau_f^2 & \frac{1}{6}\tau_f^3 & \frac{1}{12}\tau_f^4 & \frac{1}{20}\tau_f^5 & \frac{1}{30}\tau_f^6 \\ 0 & 0 & 1 & \tau_f & \frac{1}{2}\tau_f^2 & \frac{1}{3}\tau_f^3 & \frac{1}{4}\tau_f^4 & \frac{1}{5}\tau_f^5 \\ 0 & 0 & 0 & 1 & \tau_f & \tau_f^2 & \tau_f^3 & \tau_f^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = \begin{bmatrix} x_0 \\ x_0' \\ x_0'' \\ x_0''' \\ x_f \\ x_f' \\ x_f'' \\ x_f''' \end{bmatrix} \quad (62)$$

and using the following MATLAB codes to determine the coefficients:

```
symstaufx1x2x3x4x5x6x7x8
A = [1 0 0 0 0 0 0 0;
0 1 0 0 0 0 0 0;
     0 0 1 0 0 0 0 0;
     0 0 0 1 0 0 0 0;
     1 tauf 0.5*tauf^2 (1/6*tauf^3) (1/24*tauf^4) (1/60*tauf^5)
(1/120*tauf^6) (1/210*tauf^7);
     0 1 tauf 0.5*tauf^2 (1/6*tauf^3) (1/12*tauf^4) (1/20*tauf^5)
(1/30*tauf^6);
     0 0 1 tauf 0.5*tauf^2 (1/3*tauf^3) (1/4*tauf^4) (1/5*tauf^5);
0 0 0 1 tauf tauf^2 tauf^3 tauf^4];
B = [x1;x2;x3;x4;x5;x6;x7;x8];
pretty(inv(A)*B)
```

which yields

$$
\begin{aligned}
a_0 &= x_0 \\
a_1 &= x_0' \\
a_2 &= x_0'' \\
a_3 &= x_0''' \\
a_4 &= \frac{-4x_f''' - 16x_0'''}{\tau_f} + \frac{60x_f'' - 120x_0''}{\tau_f^2} + \frac{-360x_f' - 480x_0'}{\tau_f^3} + \frac{840x_f - 840x_0}{\tau_f^4} \\
a_5 &= \frac{30x_f''' + 60x_0'''}{\tau_f^2} + \frac{-420x_f'' + 600x_0''}{\tau_f^3} + \frac{2340x_f' + 2700x_0'}{\tau_f^4} + \frac{-5040x_f + 5040x_0}{\tau_f^5} \\
a_6 &= \frac{-60x_f''' - 80x_0'''}{\tau_f^3} + \frac{780x_f'' - 900x_0''}{\tau_f^4} + \frac{-4080x_f' - 4320x_0'}{\tau_f^5} + \frac{8400x_f - 8400x_0}{\tau_f^6} \\
a_7 &= \frac{35x_f''' + 35x_0'''}{\tau_f^4} + \frac{-420x_0''' + 420x_0''}{\tau_f^5} + \frac{2100x_f' + 2100x_0'}{\tau_f^6} + \frac{-4200x_f + 4200x_0}{\tau_f^7}
\end{aligned}
\tag{63}
$$

Likewise, we perform the above routine to determine the unknown coefficients for y and z.

Similarly for the speed profile in virtual domain

$$
V(\tau) = \lambda(\tau)\sqrt{x'(\tau)^2 + y'(\tau)^2 + z'(\tau)^2} \tag{64}
$$

We assume the speed factor can be expressed with the following reference functions:

$$
\begin{aligned}
\lambda(\tau) &= \sum_{k=0}^{n} b_k \frac{\{\max(1, k-2)\}!\,\tau^k}{k!} \\
\lambda'(\tau) &= \sum_{k=1}^{n} b_k \frac{\{\max(1, k-2)\}!\,\tau^{k-1}}{(k-1)!} \\
\lambda''(\tau) &= \sum_{k=2}^{n} b_k \tau^{k-2}
\end{aligned}
\tag{65}
$$

We set the initial and final value of λ (i.e., λ_0 and λ_f respectively) to 1, and the first order derivatives will be set to 0, while the second order derivatives at both endpoints will be used as varied parameters. This requires a polynomial function of degree $n = 5$.

Solving the following matrix of algebraic equations to determine the unknown coefficients for $\lambda(\tau)$:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & \tau_f & \frac{1}{2}\tau_f^2 & \frac{1}{6}\tau_f^3 & \frac{1}{12}\tau_f^4 & \frac{1}{20}\tau_f^5 \\ 0 & 1 & \tau_f & \frac{1}{2}\tau_f^2 & \frac{1}{3}\tau_f^3 & \frac{1}{4}\tau 4 \\ 0 & 0 & 1 & \tau_f & \tau_f^2 & \tau_f^3 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \begin{bmatrix} \lambda_0 \\ \lambda_0' \\ \lambda_0'' \\ \lambda_f \\ \lambda_f' \\ \lambda_f'' \end{bmatrix} \tag{66}$$

which yields

$$\begin{aligned} b_0 &= \lambda_0 \quad b_1 = \lambda_0' \quad b_2 = \lambda_0'' \\ b_3 &= \frac{3\lambda_f'' - 9\lambda_0''}{\tau_f} + \frac{-24\lambda_f' - 36\lambda_0'}{\tau_f^2} + \frac{60\lambda_f - 60\lambda_0}{\tau_f^3} \\ b_4 &= \frac{-12\lambda_f'' + 18\lambda_0''}{\tau_f^2} + \frac{84\lambda_f' + 96\lambda_0'}{\tau_f^3} + \frac{-180\lambda_f + 180\lambda_0}{\tau_f^4} \\ b_5 &= \frac{10\lambda_f'' - 10\lambda_0''}{\tau_f^3} + \frac{-60\lambda_f' - 60\lambda_0'}{\tau_f^4} + \frac{120\lambda_f - 120\lambda_0}{\tau_f^5} \end{aligned} \tag{67}$$

4. Numerical Computation

The final step is to solve the trajectory optimization problem numerically. This involves discretization of the virtual arc τ_f into $N-1$ equal segments as shown in Figure 63. Thus the length of each segment is:

$$\Delta\tau = \frac{\tau_f}{N-1} \tag{68}$$

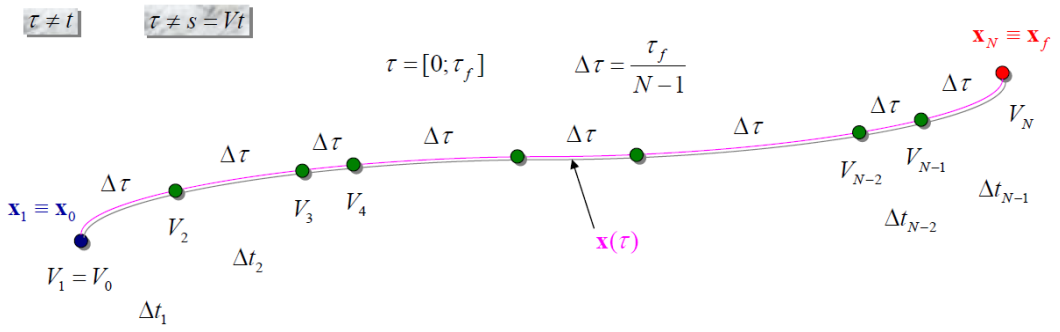


Figure 63: Excluding Time and Converting Back to Time (O. Yakimenko 2001).

All the states x and controls u at the first node $(j=1)$ are defined. For each subsequent $(j=2,3,...,N)$ node, the current value of the Cartesian coordinates $x_j(\tau_j), y_j(\tau_j), z_j(\tau_j)$ and the speed factor $\lambda_j(\tau_j)$ are computed.

The time passed since the last node is given as:

$$\Delta t_{j-1} = \frac{2\sqrt{\left(x_j - x_{j-1}\right)^2 + \left(y_j - y_{j-1}\right)^2 + \left(z_j - z_{j-1}\right)^2}}{V_j + V_{j-1}} \tag{69}$$

and the current time is:

$$t_j = t_{j-1} + \Delta t_{j-1} \tag{70}$$

The current value of the speed factor

$$\lambda_j = \frac{\Delta\tau}{\Delta t_{j-1}} \tag{71}$$

The τ derivatives can now be converted back to time domain using the following chain rule:

$$\dot{x} = \frac{dx}{dt} = \frac{dx}{d\tau}\frac{d\tau}{dt} = x'\lambda$$
$$\ddot{x} = \frac{d^2x}{dt^2} = \frac{d}{dt}\left(x'\lambda\right) = \left(x''\lambda + x'\lambda'\right)\lambda$$
$$\dddot{x} = \frac{d^3x}{dt^3} = \frac{d}{dt}\left\{\left(x''\lambda + x'\lambda'\right)\lambda\right\} \qquad (72)$$
$$== \lambda'\left(x''\lambda + x'\lambda'\right) + \lambda\left(2x''\lambda' + \lambda x''' + x'\lambda''\right)$$

and so on

In the same manner, we can find $\dot{y}, \ddot{y}, \dddot{y}, \ldots$ and $\dot{z}, \ddot{z}, \dddot{z}, \ldots$

Having computed the Cartesian coordinates and the speed factor, we can then substitute these coordinates back to Eqn.(16) and Eqn.(48) to determine the remaining states and controls.

D. TRAJECTORY FOLLOWING CONTROL LAW

The controller for the Qball-X4 quadrotor are designed to operate near hover conditions and the PID controller of the quadrotor autopilot for $\dot{z}$ control is such that:

$$\dot{z} + g > 0 \qquad (73)$$

To follow the optimal trajectory, the control input to maneuver the quadrotor in the horizontal plane is given as (Cichella, et al. 2012):

$$\begin{bmatrix} \phi_c \\ \theta_c \end{bmatrix} = \frac{1}{\dot{z} + g}\begin{bmatrix} \cos\psi & \sin\psi \\ -\sin\psi & \cos\psi \end{bmatrix}\begin{bmatrix} \ddot{x}_d + k_p\left(x_d - x\right) + k_d\left(\dot{x}_d - \dot{x}\right) \\ \ddot{y}_d + k_p\left(y_d - y\right) + k_d\left(\dot{y}_d - \dot{y}\right) \end{bmatrix} \qquad (74)$$

where, the left hand side of Eqn.(74) represents the commanded roll and pitch angles. Terms with subscript *d* on the right hand side represent the desired acceleration,

position and velocities generated from the optimal trajectory generator.

THIS PAGE INTENTIONALLY LEFT BLANK

VIII. SIMULINK IMPLEMENTATION OF THE IDVD METHOD

A. MISSION SCENARIO

A mission scenario was established to validate the IDVD direct method. In this scenario, the quadrotor was to navigate from $z = -1.5\,m$ to $z = 1.5\,m$ at a height of $1.0\,m$, with an $0.5\,m \times 0.5\,m \times 2.0\,m$ obstacle placed at the origin. Figure 64(a) shows the isometric view of the mission scenario, and Figure 64(b) shows its plan view.

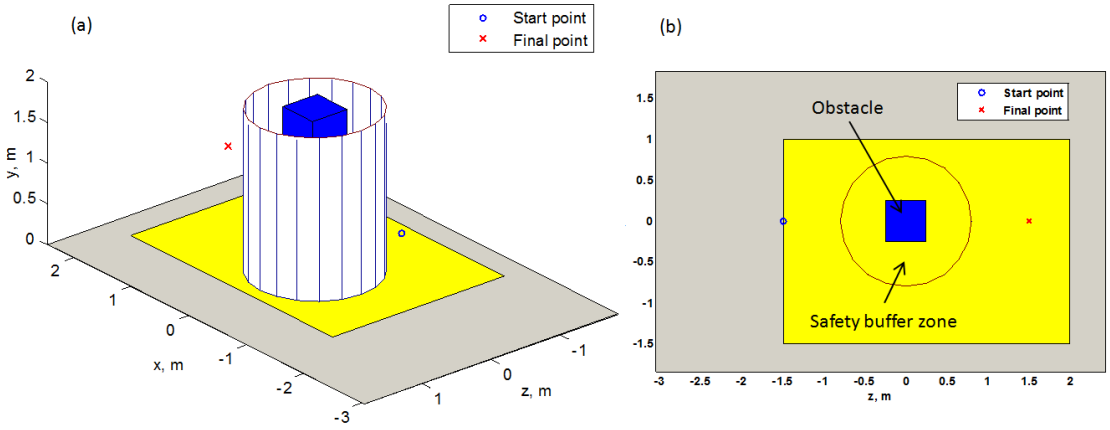


Figure 64: Obstacle Collision Avoidance Mission Scenario.

B. SIMULINK IMPLEMENTATION

In general, there are two steps in implementing the IDVD method: generating the trajectory and interfacing with the controllers used for following the trajectory. The procedure of implementing these two steps in the developed 6DOF simulation model is described in this section.

1. Trajectory Generator

Based on the methodology described earlier in Chapter 0, the rapid prototyping of the quasi-optimal trajectory generator was developed in the Simulink modeling environment. The optimization script used for generating the trajectory is presented in Appendix D. An overview of the model used for implementing the algorithm is shown in Figure 65.

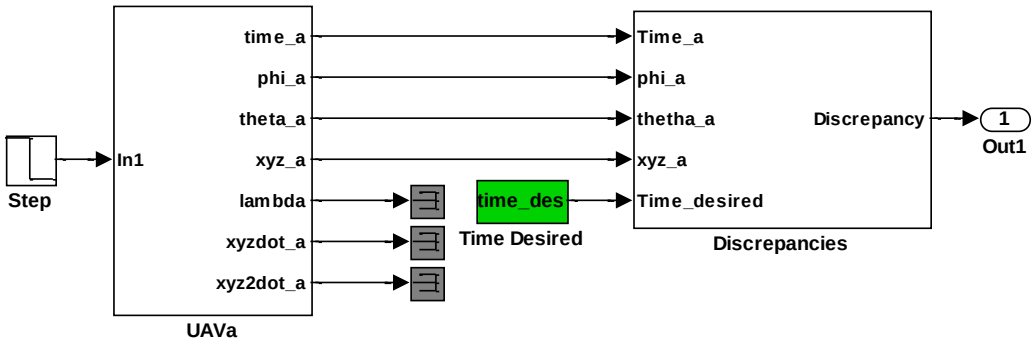


Figure 65: Overview of the Optimal Trajectory Generator.

Figure 66 shows the location at which the initial and final boundary conditions to be satisfied are set in the Simulink model. The model also computes the unknown coefficients of Eqn.(63) for the Cartesian coordinates (x, y, z) and Eqn.(67) for the speed profile. The model computes all states in the time domain as well. The outputs are the Cartesian coordinates, velocities and accelerations of the quasi-optimal trajectory, as well as the reference pitch and roll angles. Two hundred sub-intervals were defined for the optimal trajectory generated.

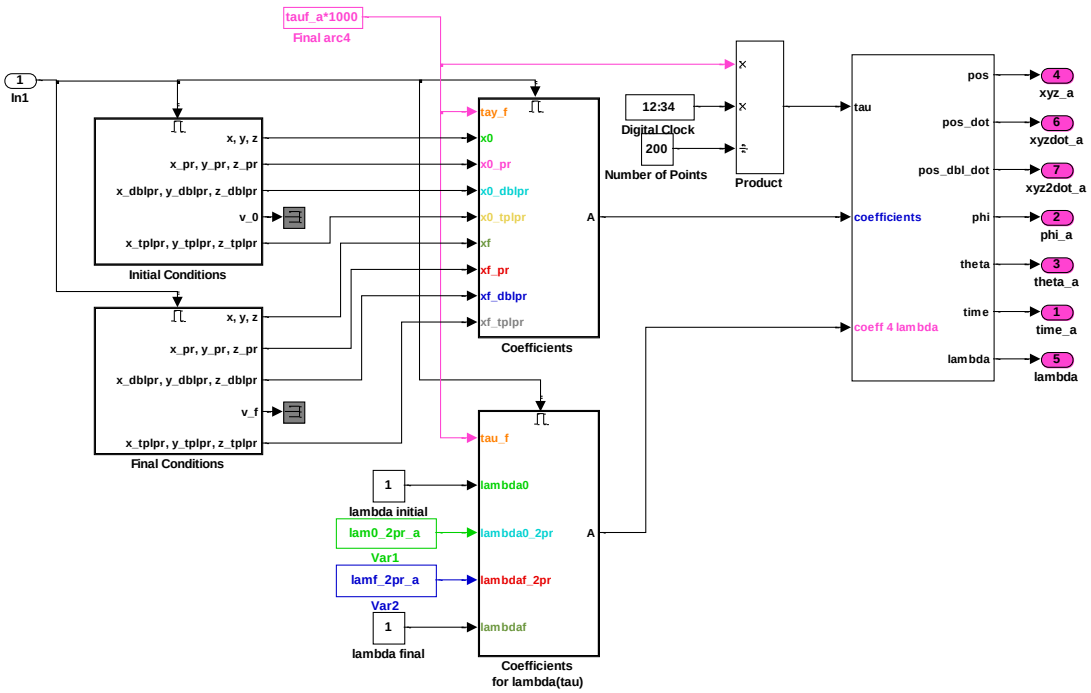


Figure 66: Implementation of IDVD Optimization Algorithms.

The discrepancies block ensures dynamics and control constraints satisfaction by the quadrotor. Also, the space (obstacle and laboratory space) constraints and desired time of arrival are also set in this block shown in Figure 67. Higher weights are assigned in the performance index for meeting the desired time of arrival and obstacle avoidance, while smaller weights are given to the quadrotor dynamics and the laboratory space constraints.

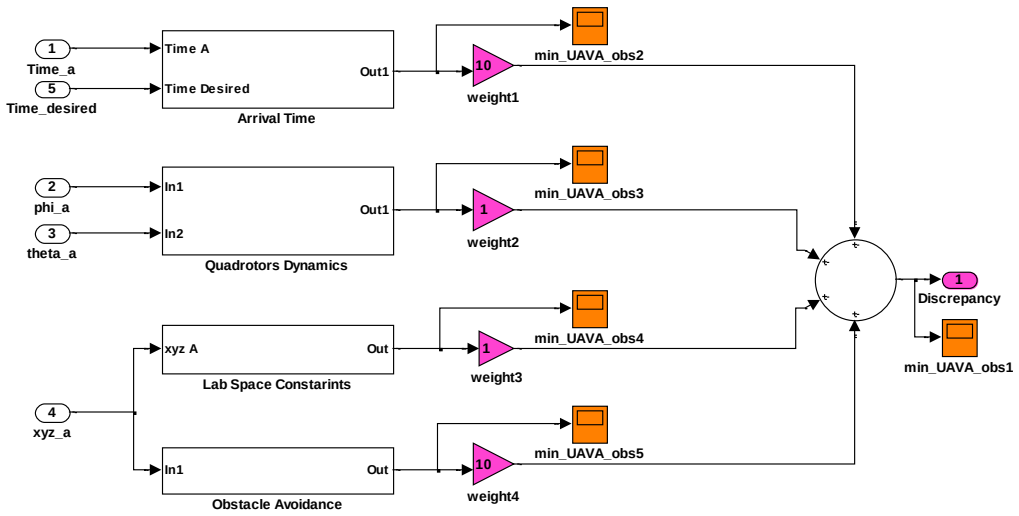


Figure 67: Discrepancies Block.

2. Trajectory Follower

The following modifications were done for the Commands subsystem module and was shown in Figure 68. The Direct Method block outputs the desired accelerations, velocities and positions in inertia frame. These information is then sent to the path follower module which use them to generate the required roll and pitch commands for the Qball-X4. The inner loop controllers are then used to track these roll and pitch commands.

There is also a switch feature which enables the operator to select either to use the Waypoint State Machine or the Direct method for guidance. 0 indicates to use the Waypoint State Machine and 1 indicates using the Direct Method. This is also being highlighted in Figure 68.

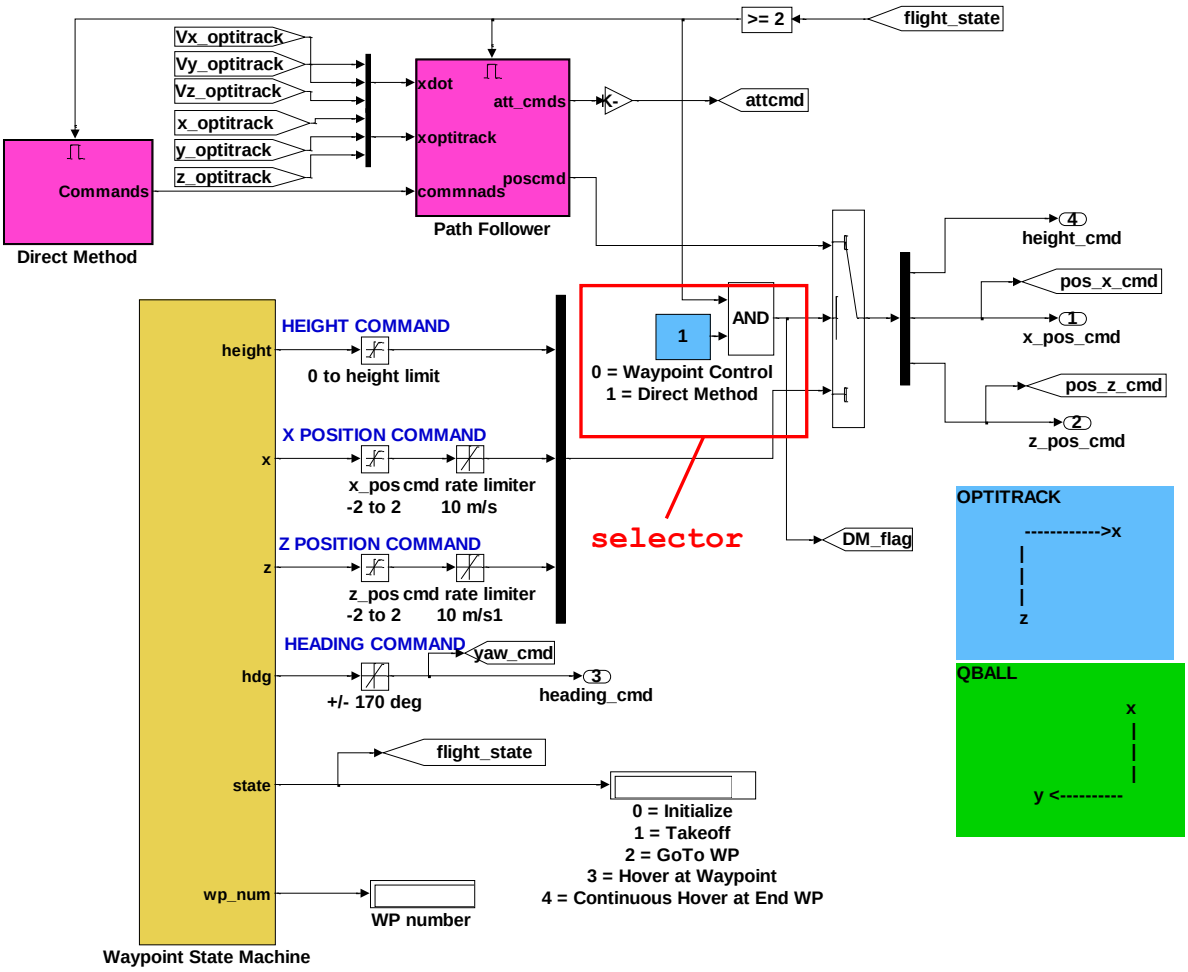


Figure 68: Modification to Controls Module to Include Optimal Trajectory Generator and Follower

C. SIMULATED RESULTS

This section shows the simulated results for the obstacle avoidance scenario.

1. Ground Track

Figure 69 shows the simulated ground track for the quadrotor. As can be seen from the figure, the quadrotor tracked the trajectory very well in the beginning but some overshoot was observed when it was near to the final

position. This can be attributed to the inner loop controller which requires tuning to improve its performance.

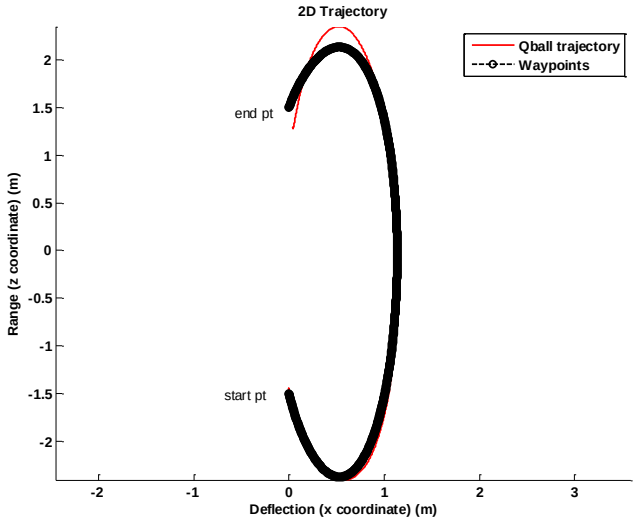


Figure 69: Ground Track (Direct Method)

2. Position Control

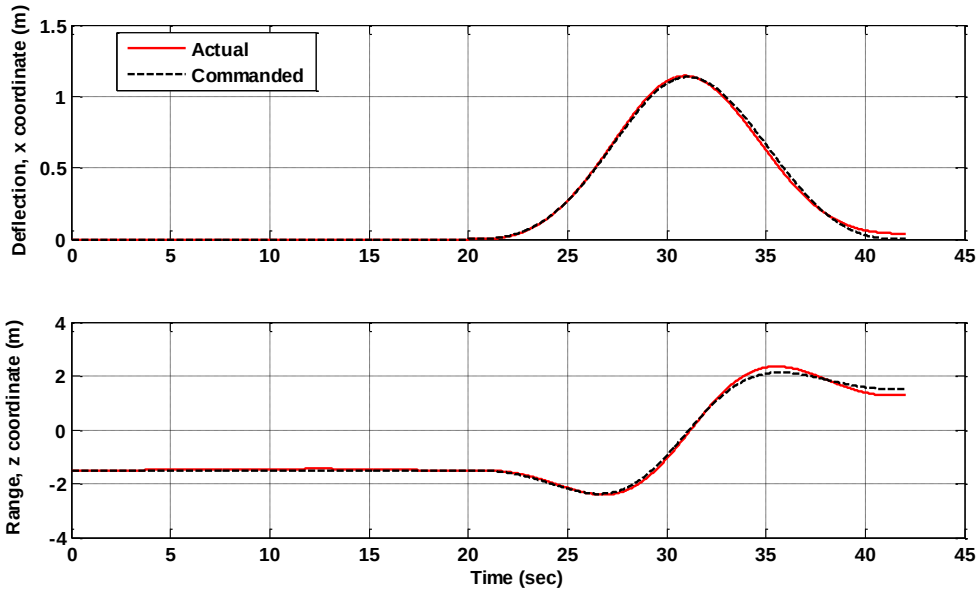


Figure 70: Position Control (Direct Method)

Figure 70 shows the performance for the position control. It can be observed that the quadrotor tracks the commanded values very closely at the beginning, but overshoot slightly during the final phase.

3. Height Control

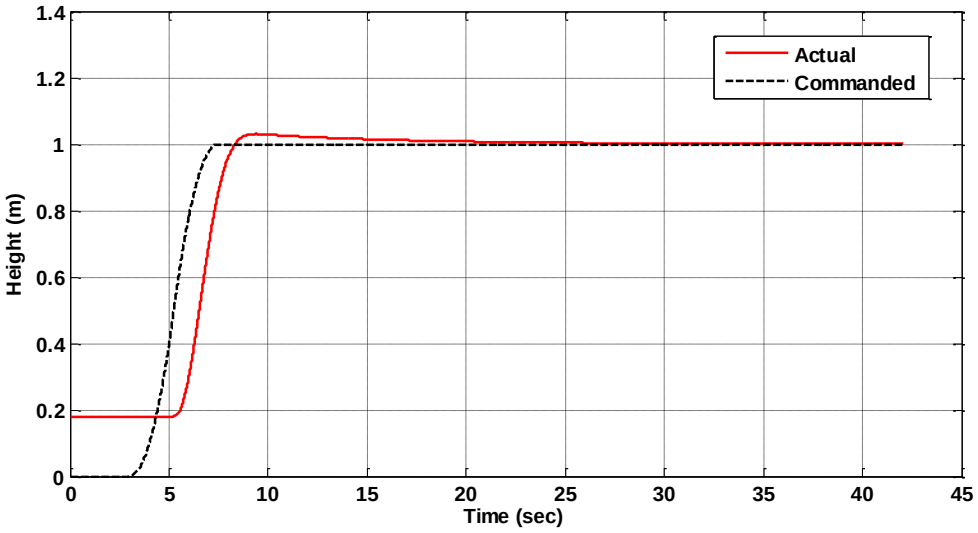


Figure 71: Height Performance (Direct Method)

Figure 71 shows the height performance of the quadrotor. It should be noted that height control is independent of the attitude (ϕ,θ,ψ) control, such that the controllers for height and attitude control are decoupled.

4. Attitude Control

Figure 72 shows the performance for the attitude control. It can be seen from the figure that the command tracking is not very good. This resulted in some overshoot in the position control. Thus, to improve the tracking

performance, the inner loop attitude controller needs to be tuned better. Unfortunately, this had not been completed due to time constraints. However, it is being proposed as a future work in the Chapter 0.

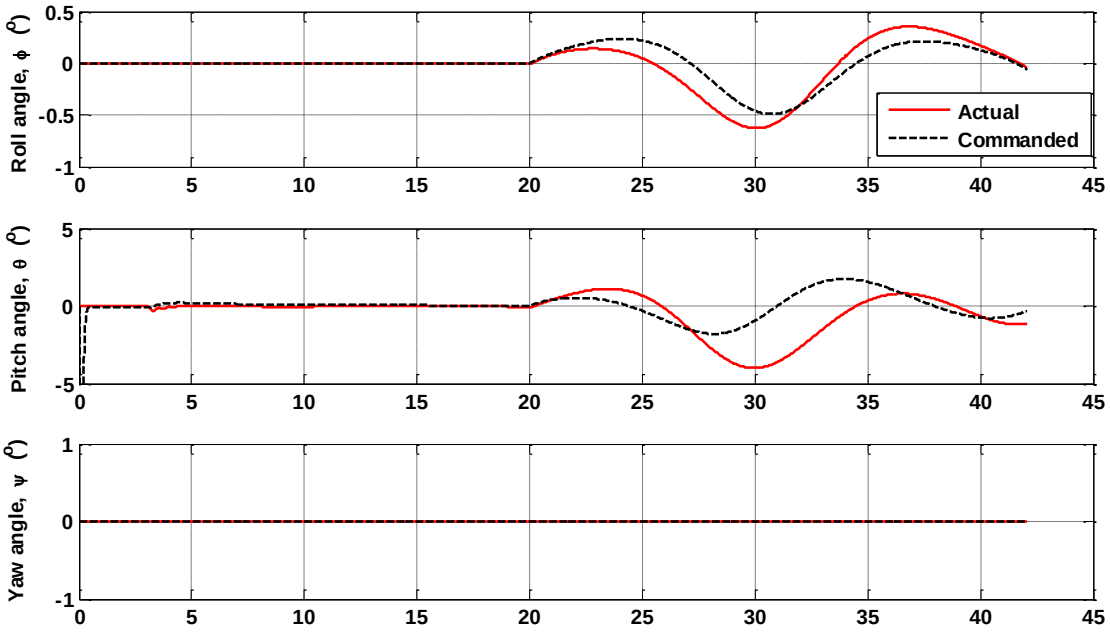


Figure 72: Attitude Control (Direct Method)

5. 3D Trajectory

Figure 73 shows the screenshot of the 3D trajectory performed by the quadrotor with an obstacle positioned in the centre between the starting and final positions.

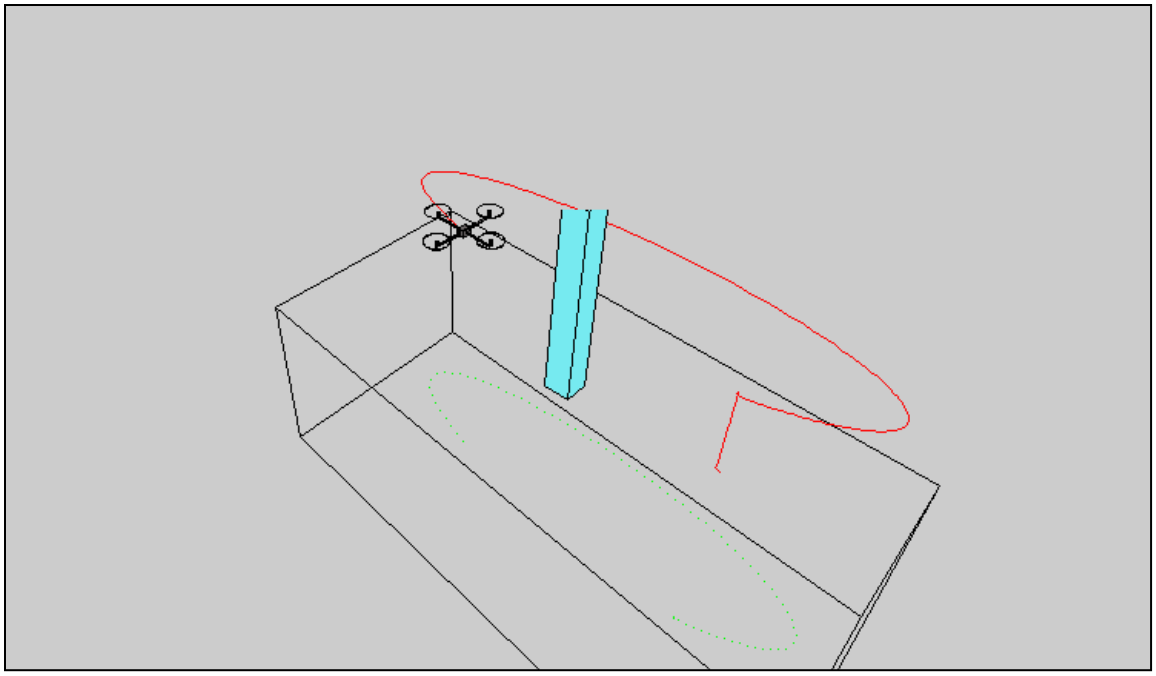


Figure 73: 3D Trajectory (Direct Method)

THIS PAGE INTENTIONALLY LEFT BLANK

IX. CONCLUSION AND FUTURE WORK

A. CONCLUSION

The following conclusions can be drawn from the research of this thesis:

- The simulated results using linearized quadrotor dynamics shows some degree of accuracy as compared to the actual results when the quadrotor flight does not deviate far from hover conditions (non-complex maneuvers).
- Results obtained from the dilution of precision (GDOP) analysis of the ASEIL lab agree with the actual test results.
- More Optitrack cameras (an additional 8 to 14 cameras) are required to be installed around the ASEIL lab in order to provide better coverage of the test workspace.
- The Inverse Dynamics in Virtual Domain (IDVD) method that depends only on a few varied parameters offers a viable solution to the Qball-X4 quadrotor, as well as any other platforms, for real-time generation of feasible trajectories.
- Path following using decoupled pitch and roll channel controllers are shown to give very poor position tracking.
- The controller based on the Lyapunov approach in SO(3) is shown to be more effective in following the desired trajectory.

B. FUTURE WORK

For the continuation of this thesis, several future efforts can be proposed as follows:

- Develop a nonlinear dynamics 6DOF model for the quadrotor to allow for more complex maneuvers by the quadrotor.
- Improve the inner loop controller of the quadrotor.
- Optimize the Optitrack cameras' location and orientation in the ASEIL lab to allow for better coverage of the test workspace.
- Implement and experiment with the path generation and path following algorithms in the actual Qball-X4 quadrotor.
- Install optical or other types of sensors onboard the Qball-X4 to allow for real-time detection of obstacles and develop codes that interface with the IDVD algorithms for real-time generation of quasi-optimal trajectories for the quadrotor to navigate around those obstacles.
- Develop the Qball-X4 fully into a field-deployable quadrotor; this requires development of the following systems:
    1. Navigation (GPS, Lidar, etc.)
    2. Communication (Datalink)
    3. Ground Control Station (GUI, software programming, etc.)

4. Guidance and Control (from launch to recovery)

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. EQUIPMENT AND LABORATORY SETUP

Details of the equipment and the application software that were used in the work of this thesis are presented in this appendix. The test setup procedures were also explained.

A. OVERVIEW

A ground control station running the host model of the Optitrack motion capture system and the Qball-X4 controller model collects localization data from a collection of 10 infrared cameras and transfer this information to the aerial vehicle via an ad-hoc wireless network.

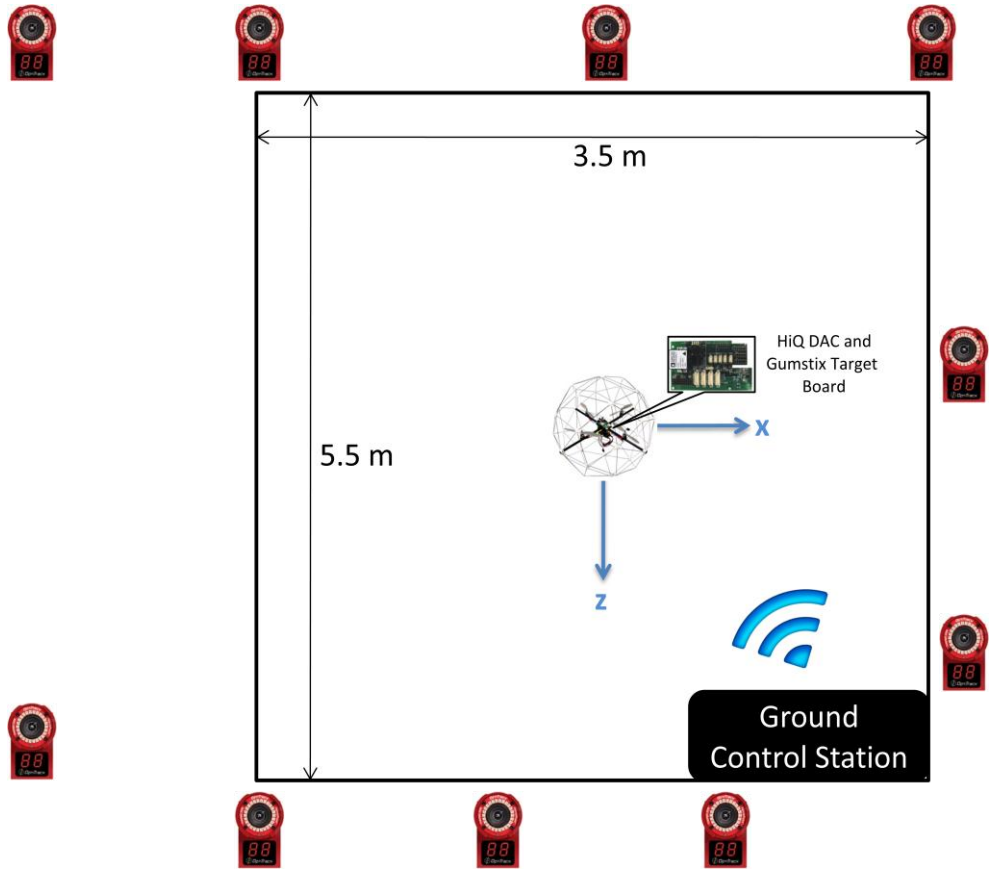


Figure 74: Laboratory Layout.

The Qball-X4 controller model was built using MATLAB/ Simulink, which was compiled and uploaded into an executable onboard the embedded Gumstix target computer wirelessly. HiQ is an embedded avionics data acquisition card (DAQ) integrated with the Gumstix Target Computer. It provides the inertial measurements and output motor commands.

B. APPLICATION SOFTWARE

1. Quanser Real-Time Control Software (QuaRC)

QuaRC version 2.2 was used for this thesis. QuaRC is a rapid prototyping and controller design and testing software developed by Quanser. The QuaRC package is used in concert with the Simulink and Real-Time workshop to allow high-level programming of the Qball-X4 controller and offers additional blockset in the Simulink library to interface with the third-party Natural Point Optitrack motion capture system. One or multiple controllers designed in Simulink can be converted into real-time executable codes via QuaRC and run on different target processors. On-line parameters tuning is also made possible through the use of the software.

2. Natural Point Tracking Tool

Natural Point Tracking Tool version 2.3.3 was used. The tool allowed for 3D marker and 6DOF object motion tracking and for calibration of the Optitrack cameras.

3. MATLAB/Simulink

MATLAB(R2011b) version 7.13 and Simulink version 7.8 were used. MATLAB is a high-level language and interactive

environment for numerical computation, visualization and programming, while Simulink toolbox offers a block diagram environment for multi-domain simulation and model-based design.

C. HARDWARE

1. Desktop Computer (Ground Control Station)

A desktop computer with the following specifications was used.

Processor:	Intel(R) Core i5 CPU @ 3.20 GHz
Operating System:	Microsoft Windows 7 Professional SP1
System Type:	32-bit
RAM:	12.0 GB

In addition, wireless communications were achieved through a wireless network adaptor inserted into the PC, while a USB 2.0 port was used for connecting the Optitrack motion capture system.

2. HiQ DAC and Gumstix Target Computer

The HiQ is the data acquisition card, which is integrated with the Gumstix target computer that runs on a Linux-based operating system. The HiQ-Gumstix functions as the IMU and flight computer for the Qball-X4 quadrotor.

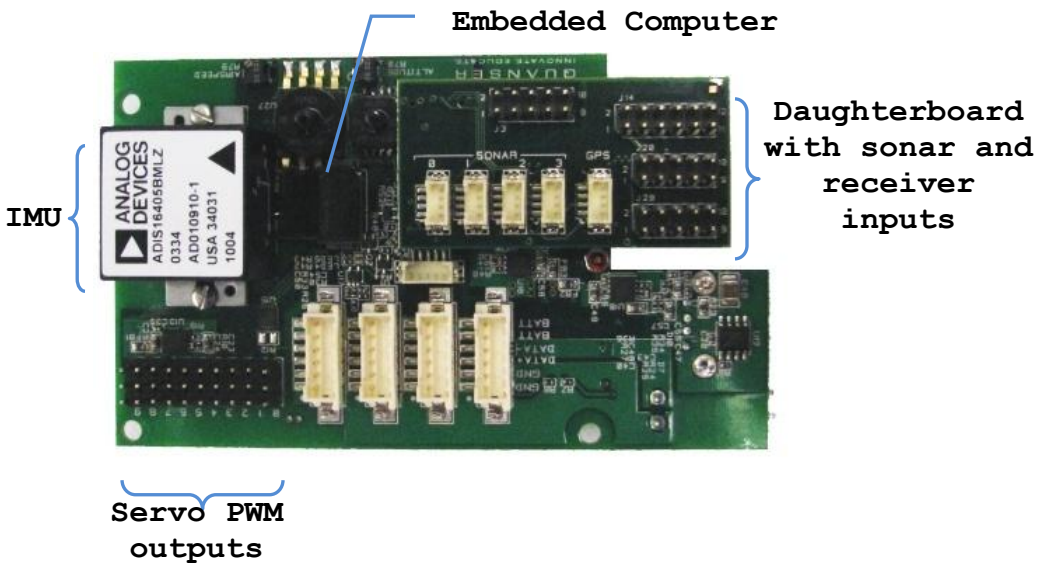


Figure 75: HiQ-embedded Avionics Data Acquisition Card.

Input/Outputs (I/Os) for the HiQ data acquisition card are the following:

- Input power 10-20 V, 400 mA typical current draw
- 10 PWM outputs (servo motor outputs)
- 6 analog inputs, 12-bit, +3.3 V
- 11 reconfigurable digital I/O
- 3-axis accelerometer, resolution 3.33 mg/LSB
- 3-axis gyroscope, range configurable for ±75°/s, ±150°/s, or ±300°/s, resolution of 0.0125°/s/LSB at range setting of ±75°/s
- 3-axis magnetometer, resolution of 0.5mGa/LSB

- 4 sonar inputs, 1 cm resolution
- TTL serial GPS input
- 2 general purpose TTL serial ports
- 8 channel RF receiver input
- USB input for onboard camera (up to 9 fps)
- 2 pressure sensors, absolute and relative

3. Optitrack Motion Capture System

The Optitrack Motion Capture System is a camera-based localization and tracking system which supports the use of at least six motion capturing infrared cameras. Multiple objects with unique marker configurations can be tracked via the reflected light from the LEDs integrated into the cameras. Ten Optitrack IR cameras were employed for the laboratory setup to track the position of the Qball-X4.

Figure 76: Natural Point Optitrack Cameras(Model V100:R2).

The features and technical specifications of the Optitrack Motion Capture System are provided as follows:

- Resolution: 640 x 480
- Frame Rate: 100 fps
- Latency: 10 ms
- Up to 16 cameras can be connected and configured for single or multiple capture volumes
- Capture volumes up to 400 feet
- Single point tracking for up to 80 markers, or 10 rigid-body objects
- Calibration time varies. Might take minutes to approximately 3 hours for high resolution optimization solution.
- Tracking accuracy on the order of mm

4. Qball-X4 Quadrotor

The Qball-X4 is a quadrotor enclosed within a patented protective carbon fiber cage. The propulsion system consists of four E-Flite Park 400 (740 Kv) motors with paired counter-rotating APC 10"×4.7" propellers. Onboard the aerial vehicle is the HiQ DAC and Gumstix embedded target computer, powered by two3-cell, 2500 mAh, Lithium Polymer batteries. The maximum endurance for the vehicle is approximately 20 min.

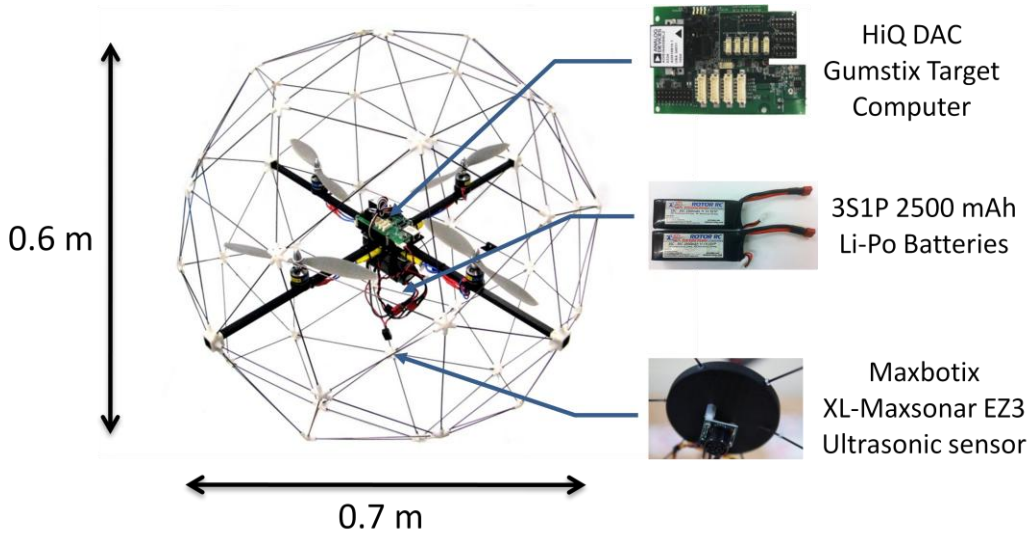
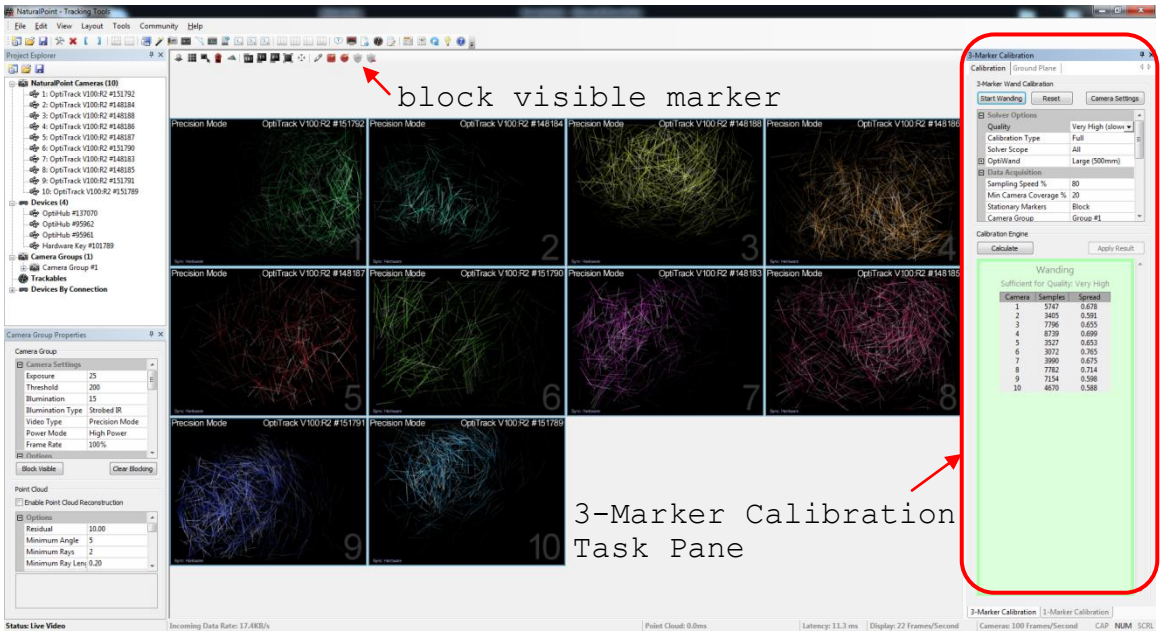


Figure 77: Dimensions of Qball-X4 and its Onboard Components.

D. SETUP PROCEDURES

Calibration of Optitrack Motion Capture System:

1. Open *Natural Point Tracking Tools* software.
2. Under "Choose a Starting Task" dialog box > Select "Perform Camera Calibration."



3. Under "3-Marker Calibration" task pane> Choose "Very High" for Quality in the Solver options.
4. Remove any reflective objects that are within the camera's field of view. If the objects cannot be physically removed, click on the "Block Visible Marker" icon that the objects are ignored during the calibration process.
5. Click on "Start Wanding" button.
6. Start swaying the calibration wand, as shown in Figure 78(a), in the space in which the aerial vehicle operates. Once sufficient data points are collected, the background color of the "Calibration Engine" task pane will appear green.

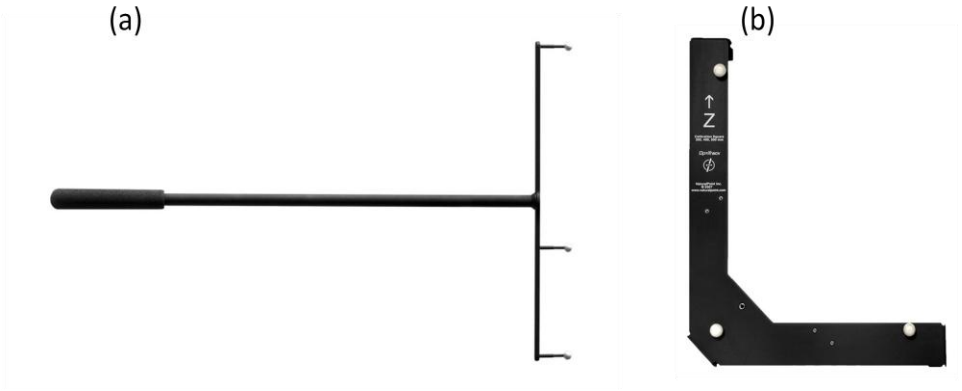


Figure 78: (a) Calibration Wand (b) Calibration Square.

7. Click on the "Calculate" button under the "Calibration Engine" task pane. Wait until the "Ready to Apply" button appears.
8. Click on "Apply Result" button.
9. Save the file (.cal file).

10. The Ground Plane Calibration screen will pop up. Use the Calibration Square as displayed in Figure 78(b), to set the (0,0,0) position of the workspace. Orient the calibration square similar to Figure 79.

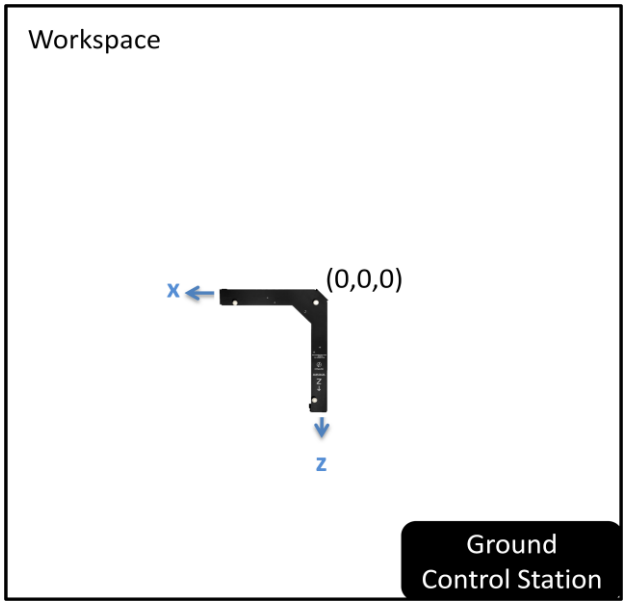


Figure 79: Orientation of Calibration Square in Workspace.

11. Save the file (.cal file) again.

12. Place the Qball-X4 with at least three attached reflective markers in the workspace. (The position of the markers must not be symmetrical, so that the orientation of the vehicle is discernible by the Optitrack motion capture system).

13. Use the mouse to select the reflective markers. Then click on "Create from Selection". Trackable *1* will be created. Rename as desired. At the same location, give the Qball-X4 an appropriate Trackable ID (i.e., 1 to 4).

14. Save the file (.tra file).

15. Exit software.

Setup Qball and Simulink Controller Model:

1. Ensure Calibration of Optitrack Motion Capture System has been performed.
2. Attached 2× Li-Po battery to the Qball-X4 and secure them tightly with the velcro straps provided.
3. Place the Qball-X4 at the (0,0,0) position in the workspace, with the colored tape pointing towards the Ground Control Station.
4. Switch On the Qball-X4.
5. Ensure that the wireless adaptor and joystick are connected to the GCS.
6. Open MATLAB/Simulink > Open the two model files
   i. Host_Joystick_TYPE_A_Optitrack_v4.mdl
   ii. qball_x4_control_v4.mdl
7. Go to Model(i), double-click "OptiTrack Measurements" block > double-click "OptiTrack Trackables" block.
8. Under "Calibration file," browse to the .cal file obtained from the Optitrack calibration process.
9. Under "Trackables definition file," browse to the .tra file obtained from the Optitrack calibration process.
10. For "Trackable IDs,"enter the assigned Trackable ID (i.e., 1 to 4) for theQball-X4.

11. Go back to Model(i), double-click "Send Joystick to Qball-X4" block > double-click "Stream Server" block.
12. Under "URI upon which to listen," ensure the port number is the same as in Model(ii). The format should look similar to

    "tcpip://localhost:18005"

    (without the quotation marks).
13. Go back to Model(i), click on "Incremental Build" icon  on the top task bar.
14. Once completed building the codes, click on "Connect to Target" icon  on the top task bar.
15. Click on the "Run" icon.
16. Confirm that the joystick is connected properly by moving the sticks and observing the signals through the scopes. Check that the trackable scope displays j1.
17. Connect to the "GSAH" wireless network.
18. Go to Model(ii), double-click "Joystick from host" block > double-click "Stream Client".
19. Under the "URI of host to which to connect," check that the URI tcpip address is synchronized to the host computer IP address. The format should look similar to

    "tcpip://182.168.1.65:18005"

    (without the quotation marks).
20. Go back to Model(ii), go to "QUARC" on the menu list > "Options..." > "Code Generation" > "Interface".

Under "MEX-file arguments, "check that the IP address matches that of the Qball-X4. The format should like similar to

'-w -d /tmp -uri %u','tcpip://182.168.1.202:17001'

with the single quotation marks, where the highlighted portion is the IP address of the Qball-X4.

Qball	IP address
A	182.168.1.202
B	182.168.1.236
C	182.168.1.235
D	182.168.1.234

21. Go back to Model(ii), click on "Incremental Build" icon  on the top task bar.
22. Once completed building the codes, click on "Connect to Target" icon  on the top task bar.
23. Click on "Run" icon.
24. Push the joystick throttle stick up to start mission.
25. Once the mission is completed or when there is a need to stop the flight, push the joystick throttle stick down to land the Qball-X4 and stop the motors.
26. Stop Simulation, and switch off the Qball-X4 power.

APPENDIX B. PLOTTING SCRIPTS FOR ANALYSIS

For 6-DOF Model Simulation:

```
%%% This script generates the following plots for the Qball 6DOF Simulator
%%% Please use the [ENABLE PLOTS] section to enable/disable which plots you
%%% would want/dont want to analyze
% 1. 2D trajectory
% 2. X and Y Position Control
% 3. Height Control and Thrust
% 4. Roll, Pitch and Yaw Control
% 5. Body-Frame Accelerations
% 6. PQR (Angular Rates)
% 7. NED Accelerations
% 8. NED Velocities
% 9. Euler Rates
% 10. True Speed
% 11. Torque

%%% ENABLE PLOTS
% This section decides which plot to enable or disable
%              1  2  3  4  5  6  7  8  9  10 11
enable_plot = [1  1  1  1  1  1  1  1  1  1  1];

%%% DATA EXTRACT
closeall
sign = -1; rad2deg = 180/pi;
time         = telemetry.time;
pos_x_cmd    = telemetry.signals.values(:,1);              ned_Az       =
telemetry.signals.values(:,16);
pos_z_cmd    = telemetry.signals.values(:,2);              ned_Vx       =
telemetry.signals.values(:,17);
yaw_cmd      = telemetry.signals.values(:,3).*rad2deg;     ned_Vy       =
telemetry.signals.values(:,18);
roll_cmd     = telemetry.signals.values(:,4).*rad2deg;     ned_Vz       =
telemetry.signals.values(:,19);
pitch_cmd    = telemetry.signals.values(:,5).*rad2deg;     pos_x        =
telemetry.signals.values(:,20);
hgt_cmd      = telemetry.signals.values(:,6);              pos_y        =
telemetry.signals.values(:,21);
accel_x      = telemetry.signals.values(:,7);              pos_z        =
telemetry.signals.values(:,22).*sign;
accel_y      = telemetry.signals.values(:,8);              phidot       =
telemetry.signals.values(:,23).*rad2deg;
accel_z      = telemetry.signals.values(:,9);              thetadot     =
telemetry.signals.values(:,24).*sign*rad2deg;
gyro_x       = telemetry.signals.values(:,10).*rad2deg;    psidot       =
telemetry.signals.values(:,25).*rad2deg;
gyro_y       = telemetry.signals.values(:,11).*rad2deg;    phi          =
telemetry.signals.values(:,26).*rad2deg;
gyro_z       = telemetry.signals.values(:,12).*rad2deg;    theta        =
telemetry.signals.values(:,27).*sign*rad2deg;
thrust_comp  =telemetry.signals.values(:,13);             psi          =
telemetry.signals.values(:,28).*rad2deg;
ned_Ax       = telemetry.signals.values(:,14);             true_speed   =
telemetry.signals.values(:,29);
ned_Ay       = telemetry.signals.values(:,15);             torque       =
telemetry.signals.values(:,30);

%% 1. 2D Trajectory
if (enable_plot(1) == 1)
figure('name','2D Trajectory');
holdon;
plot(pos_y, pos_x,'r','LineWidth',1.5);
plot(pos_x_cmd, pos_z_cmd, 'ko--','LineWidth',1.5);
title('2D Trajectory');
```

```
xlabel('Deflection (x coordinate) (m)'); ylabel('Range (z coordinate) (m)');
axisequal;
text(pos_x_cmd(1)+0.04,pos_z_cmd(1)+0.04,'start pt');
text(pos_x_cmd(end)+0.04, pos_z_cmd(end)-0.04, 'end pt');
legend('Qballtrajectory','Waypoints');
end
%% 2. X and Y Position Control
if (enable_plot(2) == 1)
figure('name','X& Y Position Control');
subplot(211)
plot(time,pos_y,'r',time,pos_x_cmd,'k--','LineWidth',1.5); grid on;
ylabel('Deflection, x coordinate (m)');
legend('Actual','Commanded',0);
subplot(212)
plot(time,pos_x,'r',time,pos_z_cmd,'k--','LineWidth',1.5); grid on;
xlabel('Time (sec)'); ylabel('Range, z coordinate (m)');
end
%% 3. Height Control and Thrust
if (enable_plot(3) == 1)
figure('name','Height Control and Thrust');
subplot(211)
plot(time,pos_z,'r',time,hgt_cmd,'k--','LineWidth',1.5); grid on;
ylabel('Height (m)');
legend('Actual','Commanded',0);
subplot(212)
plot(time,thrust_comp,'r','LineWidth',1.5); grid on;
xlabel('Time (sec)'); ylabel('Thrust (G)');
end
%% 4. Roll, Pitch and Yaw Control
if (enable_plot(4) == 1)
figure('name','Roll, Pitch and Yaw Control');
subplot(311)
plot(time,phi,'r',time,roll_cmd,'k--','LineWidth',1.5); grid on;
ylabel('Roll angle,  \phi  (^o)');
legend('Actual','Commanded',0);
subplot(312)
plot(time,theta,'r',time,pitch_cmd,'k--','LineWidth',1.5); grid on;
ylabel('Pitch angle,  \theta  (^o)');
subplot(313)
plot(time,psi,'r',time,yaw_cmd,'k--','LineWidth',1.5); grid on;
ylabel('Yaw angle,  \psi  (^o)');
end
%% 5. Body-Frame Accelerations
if (enable_plot(5) == 1)
figure('name','Body Accelerations');
holdon;
subplot(311);
plot(time,accel_x,'r','LineWidth',1.5);
ylabel('a_x (m/s^2)'); grid on;
subplot(312);
plot(time,accel_y,'r','LineWidth',1.5);
ylabel('a_y (m/s^2)'); grid on;
subplot(313);
plot(time,accel_z,'r','LineWidth',1.5);
xlabel('Time (sec)'); ylabel('a_z (m/s^2)'); grid on;
end
%% 6. PQR (Angular Rates)
if (enable_plot(6) == 1)
figure('name','Gyroscopes');
holdon;
subplot(311);
plot(time,gyro_x,'r','LineWidth',1.5); grid on;
title('PQR (Angular Rates)');
ylabel('\itp\rm  (^o/s)')
subplot(312);
plot(time,gyro_y,'r','LineWidth',1.5); grid on;
ylabel('\itq\rm  (^o/s)')
subplot(313);
plot(time,gyro_z,'r','LineWidth',1.5); grid on;
xlabel('Time (sec)'); ylabel('\itr\rm  (^o/s)')
```

```
end
%% 7. NED Accelerations
if (enable_plot(7) == 1)
figure('name','NED Accelerations');
holdon;
subplot(311);
plot(time,ned_Ax,'r','LineWidth',1.5); grid on;
ylabel('a_N (m/s^2)')
subplot(312);
plot(time,ned_Ay,'r','LineWidth',1.5); grid on;
ylabel('a_E (m/s^2)')
subplot(313);
plot(time,ned_Az,'r','LineWidth',1.5); grid on;
xlabel('Time (sec)'); ylabel('a_D (m/s^2)')
end
%% 8. NED Velocities
if (enable_plot(8) == 1)
figure('name','NED Velocities');
holdon;
subplot(311);
plot(time,ned_Vx,'r','LineWidth',1.5); grid on;
ylabel('V_N (m/s)')
subplot(312);
plot(time,ned_Vy,'r','LineWidth',1.5); grid on;
ylabel('V_E (m/s)')
subplot(313);
plot(time,ned_Vz,'r','LineWidth',1.5); grid on;
xlabel('Time (sec)'); ylabel('V_D (m/s)')
end
%% 9. Euler Rates
if (enable_plot(9) == 1)
figure('name','Euler Rates');
holdon;
subplot(311);
plot(time,phidot,'r','LineWidth',1.5); grid on;
ylabel('\phi'' (^o/s)')
subplot(312);
plot(time,thetadot,'r','LineWidth',1.5); grid on;
ylabel('\theta'' (^o/s)')
subplot(313);
plot(time,psidot,'r','LineWidth',1.5); grid on
xlabel('Time (sec)'); ylabel('\psi'' (^o/s)');
end
%% 10. True Speed
if (enable_plot(10) == 1)
figure('name','True Speed');
holdon; grid on;
plot(time,true_speed,'r','LineWidth',1.5);
title('True Speed');
xlabel('Time (sec)'); ylabel('V_t (m/s)');
end
%% 11. Torque
if (enable_plot(11) == 1)
figure('name','Torque');
holdon; grid on;
plot(time,torque,'r','LineWidth',1.5);
title('Torque');
xlabel('Time (sec)'); ylabel('Torque (N/m)');
end
```

For QBall-X4 Test Flight:

```
%%% This script generates the plots for Qball Test Flight for Post Analysis
%%% SIGNALS:
%%% 1.  pwm output (rotor 1)      32. zoptitrack
%%% 2.  pwm output (rotor 2)      33. optitrack tracking
%%% 3.  pwm output (rotor 3)      34. new (flag)
%%% 4.  pwm output (rotor 4)      35. - empty -  "z_optitrack_TF"
%%% 5.  gyro x                    36. - empty -
%%% 6.  gyro y                    37. timeout (flag)
%%% 7.  gyro z                    38. optitrack timeout (flag)
%%% 8.  accel x                   39. sonar
%%% 9.  accel y                   40. land (flag)
%%% 10. accel z                   41. throttlecmd
%%% 11. mag x                     42. height mode (mode)
%%% 12. mag y                     43. heading mode (mode)
%%% 13. mag z                     44. position mode (mode)
%%% 14. batt volt                 45. u_roll (pwm)
%%% 15. sonar                     46. u_pitch (pwm)
%%% 16. - empty-                  47. u_yaw (pwm)
%%% 17. roll comp                 48. optitrack roll
%%% 18. pitch comp                49. optitrack pitch
%%% 19. rollcmd                  50. optitrack yaw
%%% 20. pitchcmd                 51. heightcmd aft sigmoid
%%% 21. - empty -
%%% 22. roll
%%% 23. pitch
%%% 24. headingobs
%%% 25. mag heading
%%% 26. throttle joystick
%%% 27. xposcmd
%%% 28. zposcmd
%%% 29. heightcmd
%%% 30. xoptitrack
%%% 31. yoptitrack

%%% INPUT THE .MAT FILE NAME THAT YOU WANT TO ANALYZE
%%% e.g. load qball_flight_data_29-Apr-2013_15-04-34
clearall; close all;
loadqball_flight_data_13-May-2013_transfer_func

sign = -1; rad2deg = 180/pi;

runtime         = qball_data(1,:);
pwm_rotor1      = qball_data(2,:);
pwm_rotor2      = qball_data(3,:);
pwm_rotor3      = qball_data(4,:);
pwm_rotor4      = qball_data(5,:);
gyro_x          = qball_data(6,:)*rad2deg;
gyro_y          = qball_data(7,:)*rad2deg;
gyro_z          = qball_data(8,:)*rad2deg;
accel_x         = qball_data(9,:);
accel_y         = qball_data(10,:);
accel_z         = qball_data(11,:);
mag_x           = qball_data(12,:)*rad2deg;
mag_y           = qball_data(13,:)*rad2deg;
mag_z           = qball_data(14,:)*rad2deg;
batt_volt       = qball_data(15,:);
sonar           = qball_data(16,:);
optitrack_roll    = qball_data(48,:)*sign*rad2deg;
roll_comp       = qball_data(18,:)*rad2deg;
qball_data(49,:)*rad2deg;
pitch_comp      = qball_data(19,:)*rad2deg;
qball_data(50,:)*rad2deg;
roll_cmd        = qball_data(20,:)*rad2deg;
pitch_cmd       = qball_data(21,:)*rad2deg;
roll            = qball_data(22,:)*rad2deg;
pitch           = qball_data(23,:)*rad2deg;
heading_obs     = qball_data(24,:)*rad2deg;

z_optitrack       = qball_data(32,:);
optitrack_flag    = qball_data(33,:);
new_flag          = qball_data(34,:);
z_optitrack_TF    = qball_data(35,:);
% -empty-         = qball_data(36,:);
timeout_flag      = qball_data(37,:);
opti_timeout_flag = qball_data(38,:);
sonar             = qball_data(39,:);
land_flag         = qball_data(40,:);
throt_cmd         = qball_data(41,:);
height_mode       = qball_data(42,:);
heading_mode      = qball_data(43,:);
position_mode     = qball_data(44,:);
u_roll            = qball_data(45,:);
u_pitch           = qball_data(46,:);
u_yaw             = qball_data(47,:);

optitrack_pitch   =

optitrack_yaw     =

hgt_cmd_sigmoid   = qball_data(51,:);
```

```
mag_heading     = qball_data(25,:)*rad2deg;
throt_joystick  =qball_data(26,:);
x_pos_cmd       = qball_data(27,:);
z_pos_cmd       = qball_data(28,:);
height_cmd      = qball_data(29,:);
x_optitrack     = qball_data(30,:);
y_optitrack     = qball_data(31,:);

%%% FILTER DATA
%%% To find the start and end index
start_index = find(throt_joystick> 0.1);   %% START Condition: Detect throttle joystick
position > 10%
start_index = min(start_index);
end_index   = find(abs(accel_z) > 20);      %% END Condition: Detect Acceleration Z
exceeds 20 m/s^2
end_index   = min(end_index);

%% 1. 2D Trajectory

figure('name','2D Trajectory');
holdon;
plot(x_optitrack(start_index:end_index),
z_optitrack(start_index:end_index),'r','LineWidth',1.5);
plot(x_pos_cmd(start_index:end_index), z_pos_cmd(start_index:end_index), 'ko--
','LineWidth',1.5);
title('2D Trajectory');
xlabel('Deflection (x coordinate) (m)'); ylabel('Range (z coordinate) (m)');
axisequal;
text(x_pos_cmd(start_index)+0.04,z_pos_cmd(start_index)+ 0.04,'start pt');
text(x_pos_cmd(end_index)+0.04, z_pos_cmd(end_index)-0.04, 'end pt');
legend('Qballtrajectory','Waypoints');

%% 2. X and Y Position Control
figure('name','X& Y Position Control');
subplot(211)
plot(runtime(start_index:end_index),x_optitrack(start_index:end_index),'r',...
runtime(start_index:end_index),x_pos_cmd(start_index:end_index),'k--','LineWidth',1.5);
grid on;
ylabel('Deflection, x coordinate (m)');
legend('Optitrack','Commanded');
subplot(212)
plot(runtime(start_index:end_index),z_optitrack(start_index:end_index),'r',...
runtime(start_index:end_index),z_pos_cmd(start_index:end_index),'k--','LineWidth',1.5);
grid on;
xlabel('Time (sec)'); ylabel('Range, z coordinate (m)');

%% 3. Height Control and Throttle Command
figure('name','Height Control and Throttle');
subplot(211);
plot(runtime(start_index:end_index),sonar(start_index:end_index),'r',...
runtime(start_index:end_index),y_optitrack(start_index:end_index),'b',...
runtime(start_index:end_index),hgt_cmd_sigmoid(start_index:end_index),'k--
','LineWidth',1.5);
holdon; grid on;
ylabel('Height (m)');
legend('Sonar','Optitrack','Commanded');
subplot(212);
plot(runtime(start_index:end_index),throt_cmd(start_index:end_index),'r','LineWidth',1.5)
;
xlabel('Time (sec)'); ylabel('Throttle (% d.c)'); grid on;


%% 4. Roll, Pitch and Yaw Control
figure('name','Roll, Pitch and Yaw Control');
subplot(311);
plot(runtime(start_index:end_index),roll_comp(start_index:end_index),'r',...
runtime(start_index:end_index),optitrack_roll(start_index:end_index),'b',...
runtime(start_index:end_index),roll_cmd(start_index:end_index),'k--','LineWidth',1.5);
grid on;
ylabel('Roll angle,  \phi  (^o)');
```

```
legend('Qball','Optitrack','Commanded',0);
subplot(312);
plot(runtime(start_index:end_index),pitch_comp(start_index:end_index),'r',...
runtime(start_index:end_index),optitrack_pitch(start_index:end_index),'b',...
runtime(start_index:end_index),pitch_cmd(start_index:end_index),'k--','LineWidth',1.5);
grid on;
ylabel('Pitch angle,  \theta  (^o)');
subplot(313);
hdg_cmd = zeros(length(runtime),1);
plot(runtime(start_index:end_index),mag_heading(start_index:end_index),'r',...
runtime(start_index:end_index),optitrack_yaw(start_index:end_index),'b',...
runtime(start_index:end_index),hdg_cmd(start_index:end_index),'k--','LineWidth',1.5);
grid on;
%plot(runtime,heading_obs,'r',runtime,yaw_cmd,'k--','LineWidth',1.5); grid on;
ylabel('Yaw angle,  \psi  (^o)');

%% 5. ROTOR PWM (% Duty Cycle)
figure('name','pwm outputs');
holdon;
subplot(411);
plot(runtime(start_index:end_index),pwm_rotor1(start_index:end_index),'r','LineWidth',1.5
); grid on;
title('Rear Rotor'); ylabel('% d.c.');
subplot(412);
plot(runtime(start_index:end_index),pwm_rotor2(start_index:end_index),'r','LineWidth',1.5
); grid on;
title('Front Rotor'); ylabel('% d.c.');
subplot(413);
plot(runtime(start_index:end_index),pwm_rotor3(start_index:end_index),'r','LineWidth',1.5
); grid on;
title('Left Rotor'); ylabel('% d.c.');
subplot(414);
plot(runtime(start_index:end_index),pwm_rotor4(start_index:end_index),'r','LineWidth',1.5
); grid on;
title('Right Rotor'); ylabel('% d.c.'); xlabel('Time (sec)');

%% 6. PQR (Angular Rates)
figure('name','Gyroscopes');
holdon;
subplot(311);
plot(runtime(start_index:end_index),gyro_x(start_index:end_index),'r','LineWidth',1.5);
grid on;
title('PQR (Angular Rates)');
ylabel('\itp\rm  (^o/s)')
subplot(312);
plot(runtime(start_index:end_index),gyro_y(start_index:end_index),'r','LineWidth',1.5);
grid on;
ylabel('\itq\rm  (^o/s)')
subplot(313);
plot(runtime(start_index:end_index),gyro_z(start_index:end_index),'r','LineWidth',1.5);
grid on;
xlabel('Time (sec)'); ylabel('\itr\rm  (^o/s)')

%% 7. Body Accelerations
figure('name','Body Accelerations');
holdon;
subplot(311);
plot(runtime(start_index:end_index),accel_x(start_index:end_index),'r','LineWidth',1.5);
title('Body Accelerations'); ylabel('a_x (m/s^2)'); grid on;
subplot(312);
plot(runtime(start_index:end_index),accel_y(start_index:end_index),'r','LineWidth',1.5);
ylabel('a_y (m/s^2)'); grid on;
subplot(313);
plot(runtime(start_index:end_index),accel_z(start_index:end_index),'r','LineWidth',1.5);
xlabel('Time (sec)'); ylabel('a_z (m/s^2)'); grid on;

%% 8. Magnetometer
figure('name','Magnetometer');
holdon;
subplot(311);
```

```
plot(runtime(start_index:end_index),mag_x(start_index:end_index),'r','LineWidth',1.5);
title('Magnetometer'); ylabel('Mag_x (^o)'); grid on;
subplot(312);
plot(runtime(start_index:end_index),mag_y(start_index:end_index),'r','LineWidth',1.5);
ylabel('Mag_y (^o)'); grid on;
subplot(313);
plot(runtime(start_index:end_index),mag_z(start_index:end_index),'r','LineWidth',1.5);
xlabel('Time (sec)'); ylabel('Mag_z (^o)'); grid on;

%% 9. Battery Voltage
figure('name','Battery Voltage');
batt_threshold = zeros(length(runtime),1);
batt_threshold(:) = 10.6;   %% Battery threshold as stated in manual
plot(runtime(start_index:end_index),batt_volt(start_index:end_index),'r',...
runtime(start_index:end_index),batt_threshold(start_index:end_index),'k--
','LineWidth',1.5);
title('Battery Voltage'); ylabel('Volt'); grid on;
ylimits = ylim; ylim([ylimits(1)-0.1 ylimits(2)]);
xlabel('Time (sec)');
legend('battery','threshold');

%% 10. Status Flags
figure('name','Status Flags');
subplot(511);
plot(runtime(start_index:end_index),optitrack_flag(start_index:end_index),'r','LineWidth'
,1.5);
title('Optitrack Status');
subplot(512);
plot(runtime(start_index:end_index),new_flag(start_index:end_index),'r','LineWidth',1.5);
title('New data pkt');
subplot(513);
plot(runtime(start_index:end_index),timeout_flag(start_index:end_index),'r','LineWidth',1
.5);
title('Timeout');
subplot(514);
plot(runtime(start_index:end_index),opti_timeout_flag(start_index:end_index),'r','LineWid
th',1.5);
title('Optitrack Timeout');
subplot(515);
plot(runtime(start_index:end_index),land_flag(start_index:end_index),'r','LineWidth',1.5)
;
title('Land (Failure)'); xlabel('Time (sec)');

%% 11. Modes (1 = close loop, 0 = open loop)
figure('name','Control Modes');
subplot(311);
plot(runtime(start_index:end_index),position_mode(start_index:end_index),'r','LineWidth',
1.5);
title('0: Open Loop   1: Close Loop');
ylabel('Position');
subplot(312);
plot(runtime(start_index:end_index),height_mode(start_index:end_index),'r','LineWidth',1.
5);
ylabel('Height');
subplot(313);
plot(runtime(start_index:end_index),heading_mode(start_index:end_index),'r','LineWidth',1
.5);
ylabel('Heading');
xlabel('Time (sec)');

%% 12. z_optitrack before and after transfer function
figure('name','z_optitrack_pos before and after TF = 20s/(s+20)');
plot(runtime(start_index:end_index),z_optitrack(start_index:end_index),'r',runtime(start_
index:end_index),z_optitrack_TF(start_index:end_index));
title('z-optitrack before and aft of TF = 20s/(s+20)');
legend('before','after');
xlabel('Time (sec)');
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. OPTIMIZATION SCRIPT

```
closeall, clear all, clc
warningoff
D2R = pi/180;

%% Mission inputs
globaltime_des
globalobxobyrsafeattlimit
global a0XYZ a0XYZd a0XYZ2d
global afXYZ afXYZd afXYZ2d
attlimit = 5*D2R;          % attitude angle limit (rad)
obx=0.0; oby=0; rsafe=0.8; % safe radius (r obstacle is 0.4, Qball radius is 0.3)
time_des = 30;             % Tdes desired time of mission
a0XYZ   = [ -1.5;  0;  1];  % initial position for Qball A
a0XYZd  = [   0;  0;  0];  % initial velocity for Qball A
a0XYZ2d = [   0;  0;  0];  % initial acceleration for Qball A
afXYZ   = [1.5;  0;  1];  % final position for Qball A
afXYZd  = [   0;  0;  0];  % final velocity for Qball A
afXYZ2d = [   0;  0;  0];  % final acceleration for Qball A

%% Initial Guess for varied parameters
x0=[0.015           % lam0_2pr_a
    0.015           % lamf_2pr_a
    0.1             % X0a_tpl_prime
    135*D2R         % X0a_tpl_prime_angle, radians (0 deg - Pointing North)
    0.1             % Xfa_tpl_prime
    -135*D2R        % Xfa_tpl_prime_angle, radians (0 deg - Pointing North)
time_des/1000];       % tauf_a

%% Optimization
t = cputime;
options=optimset('TolFun',1e-1,'TolX',1e-1,'Display','iter'); %,'MaxIter',1000);
%options=optimset('TolFun',1e-1,'TolX',1e-1,'Display','final');
[x0,fval,exitflag,output] = fminsearch(@DMlfun,x0,options)
%[x0,fval,exitflag,output]=fminunc(@DMlfun,x0,options)
time_elapsed       = cputime - t
    lam0_2pr_a     = x0(1);
    lamf_2pr_a     = x0(2);
X0a_tpl_prime  = x0(3);
    X0a_tpl_primeA = x0(4);
Xfa_tpl_prime  = x0(5);
Xfa_tpl_primeA = x0(6);
tauf_a         = x0(7);

%% Do a single run to record all parameters
sim('DM3', [0 200])
    time_a         = a(:,1);
    phi_a          = a(:,2);
    theta_a        = a(:,3);
    x_a            = a(:,4);
    y_a            = a(:,5);
    z_a            = a(:,6);
    lambda_a       = a(:,7);
    x_vel_a        = a(:,8);
    y_vel_a        = a(:,9);
    z_vel_a        = a(:,10);
    x_accel_a      = a(:,11);
y_accel_a      = a(:,12);
z_accel_a      = a(:,13);

%% Interpolate data between points at the same frequency the controller runs at
ctrl_t_step = .005; % Controller speed
    [m_a,n_a]   = size(a);
t_a_end     = a(m_a,1);
t_a         = 0:ctrl_t_step:t_a_end;
```

```
phi_a     = interp1(time_a,phi_a,t_a,'pchip');
theta_a   = interp1(time_a,theta_a,t_a,'pchip');
x_a       = interp1(time_a,x_a,t_a,'pchip');
y_a       = interp1(time_a,y_a,t_a,'pchip');
z_a       = interp1(time_a,z_a,t_a,'pchip');
x_vel_a   = interp1(time_a,x_vel_a,t_a,'pchip');
y_vel_a   = interp1(time_a,y_vel_a,t_a,'pchip');
z_vel_a   = interp1(time_a,z_vel_a,t_a,'pchip');
x_accel_a = interp1(time_a,x_accel_a,t_a,'pchip');
y_accel_a = interp1(time_a,y_accel_a,t_a,'pchip');
z_accel_a = interp1(time_a,z_accel_a,t_a,'pchip');

%% Plot all data
clear a; close all
[X,Y,Z] = cylinder(rsafe,20);
X=X+obx; Y=Y+oby; Z=Z*2; % obstacle data
figure% 3D projection
plot3(x_a(1),y_a(1),z_a(1),'bo'); hold on;
plot3(x_a(end),y_a(end),z_a(end),'rx')
%plot3(x_a,y_a,z_a,'b-','LineWidth',2)
legend('Start point','Final point',0)
%legend('Obstacle','Startpoint','Final point','Qball trajectory',0)
mesh(X,Y,Z), hold
patch([-0.25 0.25 0.25 -0.25]', [-0.25 -0.25 0.25 0.25]',[2 2 2 2]','b')
patch([-0.25 0.25 0.25 -0.25]', [-0.25 -0.25 0.25 0.25]',[0 0 0 0]','b')
patch([-0.25 0.25 0.25 -0.25]', [-0.25 -0.25 -0.25 -0.25]',[0 0 2 2]','b')
patch([0.25 0.25 0.25 0.25]',  [-0.25 0.25 0.25 -0.25]',[0 0 2 2]','b')
patch([0.25 -0.25 -0.25 0.25]', [0.25 0.25 0.25 0.25]',[0 0 2 2]','b')
patch([-0.25 -0.25 -0.25 -0.25]', [0.25 -0.25 -0.25 0.25]',[0 0 2 2]','b')

patch([2.43 2.43 -3.05 -3.05 2.03]',[1.84 -1.84 -1.84 1.84 1.84]',...
                                        0.01+zeros(5,1),[0.83 0.82 0.78])
patch([2 2 -1.5 -1.5 2]',[1 -1.5 -1.5 1 1]',0.02+zeros(5,1),'y')
axis([-3 3 -2 2 0 2]), axis equal
xlabel('x, m'), ylabel('y, m'),zlabel('z, m')
view([-130 25])

figure% 2D projection
plot3(x_a(1),y_a(1),z_a(1),'bo'); hold on;
plot3(x_a(end),y_a(end),z_a(end),'rx')
%plot3(x_a,y_a,z_a,'b-','LineWidth',2)
legend('Start point','Final point',0)
mesh(X,Y,Z), hold
patch([-0.25 0.25 0.25 -0.25]', [-0.25 -0.25 0.25 0.25]',[2 2 2 2]','b')
patch([-0.25 0.25 0.25 -0.25]', [-0.25 -0.25 0.25 0.25]',[0 0 0 0]','b')
patch([-0.25 0.25 0.25 -0.25]', [-0.25 -0.25 -0.25 -0.25]',[0 0 2 2]','b')
patch([0.25 0.25 0.25 0.25]',  [-0.25 0.25 0.25 -0.25]',[0 0 2 2]','b')
patch([0.25 -0.25 -0.25 0.25]', [0.25 0.25 0.25 0.25]',[0 0 2 2]','b')
patch([-0.25 -0.25 -0.25 -0.25]', [0.25 -0.25 -0.25 0.25]',[0 0 2 2]','b')
patch([2.43 2.43 -3.05 -3.05 2.03]',[1.84 -1.84 -1.84 1.84 1.84]',...
                                          0.01+zeros(5,1),[0.83 0.82 0.78])
    patch([2 2 -1.5 -1.5 2]',[1 -1.5 -1.5 1 1]',0.02+zeros(5,1),'y')
axis([-3 3 -2 2 0 2]), axis equal
xlabel('x, m'), ylabel('y, m'), zlabel('z, m')
view(2)

figure% attitude vs time
subplot(211)
plot(t_a,phi_a/D2R), hold on
plot([time_a(1) time_a(end)],attlimit/D2R*[1 1],'r--')
legend('Qball','Limitations',0)
plot([time_a(1) time_a(end)],-attlimit/D2R*[1 1],'r--')
xlabel('Time, s'), ylabel('\phi, ^o')
subplot(212)
plot(t_a,theta_a/D2R), hold on
plot([time_a(1) time_a(end)],attlimit/D2R*[1 1],'r--')
plot([time_a(1) time_a(end)],-attlimit/D2R*[1 1],'r--')
xlabel('Time, s'), ylabel('\theta, ^o')

figure% velocities vs time
```

```
subplot(311)
plot(t_a,x_vel_a)
xlabel('Time, s'), ylabel('V_x, m/s')
subplot(312)
plot(t_a,y_vel_a)
xlabel('Time, s'), ylabel('V_y, m/s')
subplot(313)
plot(t_a,z_vel_a)
xlabel('Time, s'), ylabel('V_z, m/s')

figure% speed and lambda vs time
subplot(211)
plot(t_a,sqrt(x_vel_a.^2+y_vel_a.^2+z_vel_a.^2))
xlabel('Time, s'), ylabel('V, m/s')
subplot(212)
plot(time_a,lambda_a), hold on
plot(time_des*[1 1],[1 1.2],'r--')
legend('Qball','Desired time',0)
xlabel('Time, s'), ylabel('\lambda')

%% Setup data for use in controller
% Setup a series of commands for the first waypoint
t_start = 20; %Start time for maneuver
t_a = t_a+t_start;
t_beginning = 0:ctrl_t_step:t_start-ctrl_t_step;
z_comp = ones(1,length(t_beginning));
t_comp_a = [t_beginning' t_beginning';t_a' t_a'];
x_command_a = [t_beginning' x_a(1)*z_comp';t_a' x_a'];
y_command_a = [t_beginning' y_a(1)*z_comp';t_a' y_a'];
z_command_a = [t_beginning' z_a(1)*z_comp';t_a' z_a'];
theta_command_a = [t_beginning' theta_a(1)*z_comp'; t_a' theta_a'];
phi_command_a = [t_beginning' phi_a(1)*z_comp'; t_a' phi_a'];
```

THIS PAGE INTENTIONALLY LEFT BLANK

BIBLIOGRAPHY

[1] Kansas State University. October 18, 2012. http://www.salina.k-state.edu/aviation/uas/uavs.html (accessed April 25, 2013).

[2] J. Croft, *Flightglobal*. August 17, 2010. http://www.flightglobal.com/news/articles/honeywell-readies-t-hawk-incremental-and-leap-upgrades-346118/ (accessed April 26, 2013).

[3] Net Resources International. *army-technology.com*. 2012. http://www.army-technology.com/projects/honeywell-thawk-mav-us-army/ (accessed April 26, 2013).

[4] E. Altug, J. P. Ostrowski, and R. Mahony. "*Control of a quadrotor helicopter using visual feedback*.", Proceedings of the 2002 IEEE International Conference on Robotics and Automation. Washington D.C., 2002. 72-77.

[5] Bouabdallah, Samir, A. Noth, and R. Siegwart. "*PID vs LQ control techniques applied to an indoor micro Quadrotor*." *International Conference on Intelligent Robots and Systems*. 2004.

[6] T. Bresciani. "*Modelling, Identification and Control of a Quadrotor helicopter.*" Master's Thesis, Sweden: Lund University, 2008.

[7] Bristeau, Pierre-Jean, F. Callou, D. Vissiere, and N. Petit. "*The navigation and control technology inside the AR. Drone micro UAV*." 18th IFAC World Congress. Milano: International Federation of Automatic Control, 2011.

[8] P. Chaudhari, "Aggressive maneuvers using differential flatness for a Quadrotor." Massachusetts Institute of Technology, December 12, 2011.

[9] J. Colorado, and A. Barrientos. "Mini-quadrotor attitude control based on hybrid backstepping & Frenet-Serret theory." Robotics and Automation (ICRA), 2010 IEEE International Conference. Anchorage, 2010.

[10] I. D. Cowling, J. F. Whidborne, and A. K. Cooke. "*MBPC for autonomous operation of a Quadrotor air vehicle*."

Proc 21st International UAV Systems Conference. Bristol, 2006.

[11] I. D. Cowling, O. A. Yakimenko, J. F. Whidborne, and A. K. Cooke. "*Direct method based control system for an autonomous Quadrotor.*" Journal of Intelligent and Robotic Systems, 2010: 285–316.

[12] C. Coza. "*A new robust Adaptive-Fuzzy control method applied to Quadrotor helicopter stabilization*." Fuzzy Information Processing Society, 2006. NAFIPS 2006. Annual meeting of the North American. Montreal, 2006. 454 - 458.

[13] Cutler, Mark, and J. P. How. "*Actuator constrained trajectory generation and control for variable-pitch Quadrotor.*", Massachusetts Institute of Technology, 2012.

[14] Hoffmann, M. Gabriel, and S. L. Waslander. "*Quadrotor helicopter trajectory tracking control.*" AIAA Guidance, Navigation and Control Conference and Exhibit. Hawaii, 2008.

[15] Madani, Tarek, and A. Benallegue. "*Backstepping control with exact 2-Sliding mode estimation for a Quadrotor Unmanned Aerial Vehicle.*" Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems. San Diego, 2007.

[16] Mellinger, Daniel, and V. Kumar. "*Minimum snap trajectory generation and control for Quadrotors.*" Robotics and Automation (ICRA), 2011 IEEE International Conference. Shanghai, 2011. 2520 - 2525.

[17] A. L. Salih, M. Moghavvemi, H. A. F. Mohamed, and K. Sallom. "*Flight PID controller design for a UAV quadrotor.*" Scientific Research and Essays, 2010: 3660-3667.

[18] J. F. Shepherd, and K. Tumer. "*Robust neuro-control for a micro Quadrotor.*" Genetic and Evolutionary Computation Conference 2010. Portland, 2010. 1131-1138.

[19] G. Szafranski, and R. Czyba. "*Different approaches of PID control UAV type Quadrotor.*" International Micro Air Vehicles. 2011.

[20] P. Pounds, R. Mahony, and P. Corke. "*Modeling and control of a large Quadrotor robot.*", Yale University, 2010.

[21] A. Bradford, A. Nelson, and J. Palm. "*System modeling and dynamics of the Drafanflyer XP Quadrotor UAV.*" 2010. http://depts.washington.edu/soslab/mw/images/e/e1/Group3_Spring_2010_MS2_Presentation.pdf \

(accessed January 5, 2013).

[22] I. E. Putro, A. Budiyono, and K. J. Yoon. "*Nonlinear modeling of Quadrotor Aerial Vehicle.*" Konkuk University, 2011.

[23] Goela, Rahul, S. M. Shahb, N. K. Guptac, and N. Ananthkrishnanc. "*Modeling, simulation and flight testing of an autonomous Quadrotor.*" International Conference on Environmental and Agriculture Engineering. International Conference on Environmental and Agriculture Engineering, 2009.

[24] T. Li. "*Nonlinear and fault-tolerant control techniques for a Quadrotor Unmanned Aerial Vehicle.*" Master's Thesis, Montreal, 2011.

[25] Y. Zhang, and A. Chamseddine. "*Fault tolerant flight control techniques with application to a Quadrotor UAV testbed.*" In Automatic Flight Control Systems - Latest Developments, by Thomas Lombaerts, 199-151. InTech, 2012.

[26] Office of the Secretary of Defense. *Unmanned aircraft systems.* 2005. http://www.acq.osd.mil/usd/Roadmap (accessed January 15, 2013).

[27] N. Slegers, and O. Yakimenko. "*Terminal guidance of autonomous parafoils in high wind-to-airspeed ratios.*" Journal of Aerospace Engineering, 2011: 225-336.

[28] O. Yakimenko, J. A. Lukacs IV. "*Trajectory-shaping guidance for interception of ballistic missiles during the boost phase.*" Journal of Guidance, Control and Dynamics, 2008: 1524-1531.

[29] A. Rao. "*A survey of numerical methods for optimal control.*" AAS/AIAA Astrodynamics Specialist Conference. Pittsburgh, 2009.

[30] S. Bennett. *"A History of Control Engineering 1930-1955."* London: Peter Peregrinum Ltd, 1993.

[31] A. Serirojanakul. "*Optimal control of quad-rotor helicopter using state feedback LPV method.*" Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2012 9th International Conference. Phetchaburi: ECTI-CON, 2012. 1-4.

[32] Nuchkrua, Thanana, and M. Parnichkun. "*Identification and optimal control of Quadrotor.*" *Thammasat International Journal of Science and Technology*, 2012.

[33] Hehn, Markus, and R. D'Andrea. "*Real-time trajectory generation for interception maneuvers with Quadrocopters.*" Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference. Vilamoura, 2012. 4979 - 4984.

[34] L. S. Lasdon, S. K. Mitter, and A. D. Waren. "*The conjugate gradient method for optimal control problems*." IEEE Transactions on Automatic Control, 1967: 132-138.

[35] P. R. Turner and E. Huntley. "*Self-scaling variable metric methods in hilbert space with applications to control problems.*" Optimal Control Applications and Methods, 1: 155-166, 1980.

[36] A. Cotter, O. Shamir, N. Srebro, and K. Sridharan. *"Better mini-batch algorithms via Accelerated Gradient Methods."* Cornell University, 2011.

[37] Shor, Z. Naum. *"Minimization methods for non-differentiable functions."* Springer Berlin Heidelberg, 1985.

[38] A. I. Chen and A. Ozdaglar. *"A fast distributed Proximal-Gradient method."* Cornell University, 2012.

[39] X. Chen, Z. Nashed, and L. Qi. "*Smoothing methods and semismooth methods for nondifferentiable operator equations*." Society for Industrial and Applied Mathematics, 2000: 1200-1216.

[40] Elhedhli, Samir, J. L. Goffin and J. P. Vial. "Nondifferentiable Optimization: Cutting Plane Methods." In Encyclopedia of Optimization, 2590-2595. Springer, 2009.

[41] Stoer, Josef, and R. Bulirsch. *"Introduction to numerical analysis."* New York: Springer-Verlag, 1980.

[42] O. V. Stryk. *Multiple Shooting Method.* April 5, 1996. http://www.sim.informatik.tu-darmstadt.de/publ/download/1993-manutec/node5.html (accessed January 20, 2013).

[43] A. Engelsone. *"Direct transcription methods in optimal control: Theory and Practice."* Master's Thesis, Raleigh: North Carolina State University, 2006.

[44] J. T. Betts. *"Practical methods for optimal control using nonlinear programming."* Philadelphia: Society for Industrial and Applied Mathematics, 2001.

[45] S. Devasia. *Inversion-based Feedforward.* http://faculty.washington.edu/devasia/Inversion.html (accessed March 15, 2013).

[46] W. V. Y. Chen, and Z. L. Tsong. "*Optimal control applications and methods*." Optimal Control Applications and Methods, 1998: 363-370.

[47] Q. Gong, W. Kang, and I. M. Ross. "*A Pseudospectral method for the optimal control of constrained feedback linearizable systems.*" *IEEE Transactions on Automatic Control*, 2006: 1115-1129.

[48] G. N. Elnagar and M. A. Kazemi. "*Pseudospectral chebyshev optimal control of constrained nonlinear dynamical systems.*" *Computational Optimization and Applications 11*, 1998: 195-217.

[49] D. A. Benson, G. T. Huntington, T. P. Thorvaldsen, and A. V. Rao. "*Direct trajectory optimization and costate estimation via an orthogonal collocation method.*" Journal of Guidance, Control and Dynamics, 2006: 1435-1440.

[50] D. A. Garg, D. A., et al. "*Direct trajectory optimization and costate estimation of finite-horizon and infinite-horizon optimal control problems using a Radau Pseudospectral Method.*" Computational Optimization and Applications, 2011: 335-358.

[51] Garg, Divya, W. W. Hager and A. V. Rao. "*Pseudospectral methods for solving infinite-horizon optimal control problems.*" Automatica, 2011: 829-837.

[52] Quanser. "*Quanser Qball-X4 user manual.*" User Manual, Markham, 2011.

[53] Xoneca. *Wikepedia.* http://en.wikipedia.org/wiki/File:Geometric Dilution Of Precision.svg (accessed June 16, 2013).

[54] A. Chelouah. "*Extensions of differential flat fields and Liouvillian systems.*" 36th IEEE Conf. Decision Contr. CA, 1997.

[55] S. D. Hanford, L. N. Long, and J. F. Horn. "*A small semi-autonomous rotary-wing Unmanned Air Vehicle (UAV).*" Infotech@Aerospace Conference. American Institude of Aeronautics and Astronautics, 2005.

[56] G. F. Andrews. "*A multiple-shooting technique for optimal control.*" Journal of Optimization Theory and Applications, 1999: 299-313.

[57] G. Arfken. *"The method of steepest descents."* Orlando: Academic Press, 1985.

[58] M. A. Athans, and P. L. Falb. *"Optimal control."* New York: McGraw-Hill, 1966.

[59] J. Betts. "*Optimal low thrust trajectories to the moon.*" Applied Dynamical Systems, 2003: 144-170.

[60] Bock, H. Georg, and K. J. Plitt. "*A multiple shooting algorithm for direct solution of optimal control.*" International Federation of Automatic Control. Budapest: International Federation of Automatic Control, 1984.

[61] Brandi, Primo, and A. Salvadori. "*On measure differential inclusions in optimal control theory.*" Rendiconti del Seminario Matematico, 1998: 69-86.

[62] F. Fahroo, and I. M. Ross. "*Pseudospectral methods for infinite-horizon optimal control problems.*" Journal of Guidance, Control and Dynamics, 2008.

[63] F. Fahroo, and M. Ross. "*Costate estimation by a Legendre Pseudospectral Method.*" Journal of Guidance, Control and Dynamics, 2001: 270-277.

[64] L. S. Pontryagin, V. G. Boltjanskiy, R. V. Gamkrelidze, and Mishenko E. F. *"The mathematical theory of optimal processes."* New York: Interscience Publishers, 1962.

[65] R. V. Mayorga, V. H. Quintana. "*A family of variable metric methods in function space, without exact line searches.*" Journal of Optimization Theory and Applications, 1980.

[66] H. Schattler, and U. Ledzwewicz. *"Geometric optimal control."* Springer, 2012.

[67] Shankar, Praveen, and R. K. Yedavalli. "*Dynamic inversion via state dependent Riccati equation approach: Application to flight vehicles*." American Institute of Aeronautics and Astronautics, 2009.

[68] V. T. Taranenko. "*Experience on application of Ritz's, Poincare's and Lyapunov's methods in solving flight dynamics problems*." Air Force Engineering Academy Press, 1968.

[69] O. Yakimenko. *Lecture Notes on Direct Methods for Rapid Prototyping of Optimal Maneuvers.* 2001.

[70] V. Cichella, et al. "*A Lyapunov-based approach for time-coordinated 3D path-following of multiple Quadrotors.*" Decision and Control (CDC), 2012 IEEE 51st Annual Conference, 2012: 1776-1781.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California